IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Maguire et al.	Atty Docket:	2221/103
Serial No:	10/821,750	Art Unit:	2623
Date Filed:	April 9, 2004	Examiner:	Ingvoldstad, B.
Invention:	Data Correlation and Analyses Tool		

FILED BY USPTO ELECTRONIC FILING SYSTEM

MAIL STOP AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. §1.131

1. We are co-inventors of the data correlation and analysis tool described and claimed in the above-identified patent application. John Maguire earned a PhD in Physics from Boston College and Terry Potter earned his PhD in System Science with a major in Computer Science from SUNY Binghamton in 1987.
2. We invented, programmed and built the data correlation and analysis system of the invention prior to July 12, 1995.
3. Computer programs for use in the system created prior to July 12, 1995 by and under our direction have been retrieved from an old floppy disk. A directory listing of these computer programs used in the method and system of the invention is submitted herewith as Exhibit A. Under the column entitled "modified" are dates that have been redacted, all of which are prior to July 12,

1995 indicating that the programs in the form as retrieved were available prior to July 12, 1995.

4. A printout of the retrieved program Modular.bas in its form as of prior to July 12, 1995 is provided herewith as Exhibit B. This printout was made from the floppy disk that had been maintained by the company unmodified since prior to July 12, 1995.

5. To facilitate examination of the above-identified patent application, we explain in the following paragraphs how our program in Exhibit B corresponds to the limitations of claim 49.

6. The program in Exhibit B was for use in conjunction with a response data collection program that collects responses and data from respondents who are reacting to viewing a stimulus stream. Commercially available ORTEK meter data collection software was used in conjunction with our system prior to July 12, 1995 to collect response data into one minute flat spreadsheet like files. An analog video stimulus signal, would first be processed through a digital converter, such as the commercially available Videologic's DVA video capture/overlay hardware, into a digital video with 32 frames per second.. The digital video is played and shown to the respondents. An initial operation of our system thus included "showing the at least one stimulus stream to one or more respondents." Response data entered by the respondents was captured by the ORTEK meter data collection software.

7. At Module1-page 33/34 in SUB LoadVideoFiles2(DBF$), the digitized stimulus stream is retrieved and partitioned into time slices that occur at 500 millisecond

intervals using form and method f_video.pnlVidFrame in conjunction with form and method f_video.MCIWnd1. The time slices in this version are created every 500 milliseconds as seen in method f_video.MCIWnd1(x).TimerFreq=500.

8. Referring now to Module1- page 33/34 in SUB LoadVideoFiles2(DBF$), the program instructions perform the step of associating stimuli in at least one stimulus stream with the time slice in which each stimulus occurs. Individual time slices are controlled by a timer which triggers Sub Timer1_Timer in the F_Video form. Time slices are used throughout the f_video form and its methods such as Sub MCIWnd1_positionchange as well as Sub PositionControls to associate the stimulus with the time slice in which each stimulus occurs. Note that CurAVI is a variable used throughout to identify which of the stimulus streams to use.
9. Referring now to page Module1-page 20-21 SUB DBOPEN(DBF$), the program associates responses to the at least one stimulus stream with the time slices in which each response is made. In this SUB DBOPEN individual responses and AVI digitized Video stimulus streams are associated with the same time slices in which each response is made. This Sub DBOpen also uses Sub LoadVideoFile2 found in Module1-page 33/34. The CurAVI variable specifies which digitized stimulus stream to use and a timer (Timer1) is used to coordinate the reading of the responses which come from the DB with the stimulus stream via form and method f_video.MCIWnd1.
10. Referring now to Module1-pages 18-19 create a new DataBase via Sub CreateNewModDB and uses the Sub ImportDIF on page 29 to store and

associate the response data to the time slices. The digitized Video stimulus is opened in the Sub DBOpen, and the individual image frames are associated and stored with the times slices in the f_Video.MCIWnd1 method found in Sub LoadVideoFiles2 on pages 33-34. These methods and subprograms perform the function of storing the associative mapping correlating each of the time slices with the stimuli and the responses. The f_Video.MCIWnd1 method uses a timer (Timer1) to coordinate the time slices of the digitized stimulus stream with the responses.

11. In further support of this showing that the analysis tool and method as claimed was created, built and functioned prior to July 12, 1995, Exhibit C is an explanatory document illustrating features of the program. The redacted revision date on the document is prior to July 12, 1995. The document was printed off of the old floppy disk on which the computer programs were found.
12. On the second page of Exhibit C, it is shown and explained that the record of responses can be accumulated and illustrated to the user. Moreover, the cursor may click anywhere in the response window time line thereby causing the displayed video to move to that point as well. This is only possible because of the associative map in which the responses were correlated to the time slices of the video stimulus stream. The program can immediately access the digital video frames associated with the particular response segment. Exhibit C thus confirms that the invention was created and working prior to July 12, 1995.
13. We each hereby declare that all statements made of our own knowledge herein are true and all statements made on information and belief are believed to be

true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application in connection with which this declaration is being submitted to the Patent and Trademark Office, or any patent issued thereon.

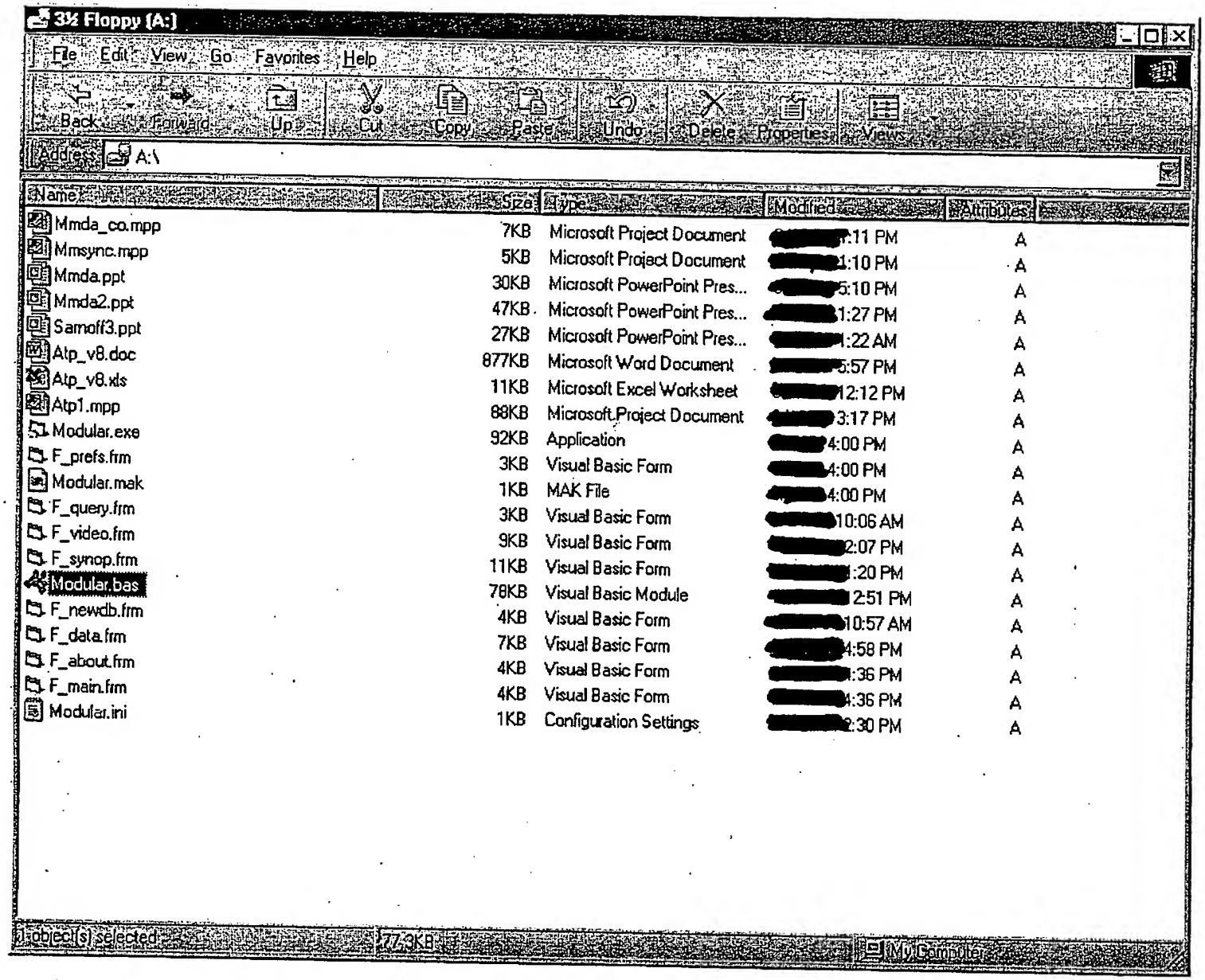
1/12/09
Date

John Maguire

1/12/09
Date

Terry Potter

02221/00103 985365.1


3½ Floppy [A:]
File Edit View Go Favorites Help
Back Forward Up Cut Copy Paste Undo Delete Properties Views
Address A:\
Name Size Type Modified Attributes
Mmda_co.mpp 7KB Microsoft Project Document [redacted]:11 PM A
Mmsync.mpp 5KB Microsoft Project Document [redacted]1:10 PM A
Mmda.ppt 30KB Microsoft PowerPoint Pres... [redacted]5:10 PM A
Mmda2.ppt 47KB Microsoft PowerPoint Pres... [redacted]1:27 PM A
Samoff3.ppt 27KB Microsoft PowerPoint Pres... [redacted]1:22 AM A
Atp_v8.doc 877KB Microsoft Word Document [redacted]5:57 PM A
Atp_v8.xls 11KB Microsoft Excel Worksheet [redacted]12:12 PM A
Atp1.mpp 88KB Microsoft Project Document [redacted]3:17 PM A
Modular.exe 92KB Application [redacted]4:00 PM A
F_prefs.frm 3KB Visual Basic Form [redacted]4:00 PM A
Modular.mak 1KB MAK File [redacted]4:00 PM A
F_query.frm 3KB Visual Basic Form [redacted]10:06 AM A
F_video.frm 9KB Visual Basic Form [redacted]2:07 PM A
F_synop.frm 11KB Visual Basic Form [redacted]:20 PM A
Modular.bas 78KB Visual Basic Module [redacted]2:51 PM A
F_newdb.frm 4KB Visual Basic Form [redacted]10:57 AM A
F_data.frm 7KB Visual Basic Form [redacted]4:58 PM A
F_about.frm 4KB Visual Basic Form [redacted]:36 PM A
F_main.frm 4KB Visual Basic Form [redacted]4:36 PM A
Modular.ini 1KB Configuration Settings [redacted]2:30 PM A
1 object(s) selected 77.3KB My Computer

Ex A

Ex. B

```
Global VArr() As String
Global XArr() As String
Global VCtr As Integer
Global VidArr() As String
Global NewDB As Database
Global TblVar As table
Global AVILength As Long
Global NumSecs As Integer
Global PrevAVITot As Long
Global NumAVIs As Integer
Global CurAVI As Integer
Global VMode As String
Global AbsPos As Long
Global TArr(25) As String
Global MyCmd As String
Global MaxX1, MaxY1, MaxX2, MaxY2, MaxX3, MaxY3 As Single
Global MinX1, MinY1, MinX2, MinY2, MinX3, MinY3 As Single
Global CurDBPath As String
Global AVILeadin As Integer
Global AVILeadout As Integer
Global GTop, GBot As Integer
Global MyStdev, MyVar As Single
Global Ds1 As snapshot
Global BarInc As Integer
Global QueryHasBeenRun As Integer
Global NewFld$
Global CancelFlag
Global DBPath$, DBFile$, DBIniFile$

'STRUCTURE SECTION --------------------

'  structures for defining DIBs
Type BITMAPINFOHEADER '40 bytes
        biSize As Long
        biWidth As Long
        biHeight As Long
        biPlanes As Integer
        biBitCount As Integer
        biCompression As Long
        biSizeImage As Long
        biXPelsPerMeter As Long
        biYPelsPerMeter As Long
        biClrUsed As Long
        biClrImportant As Long
End Type

'Type AVIFILEINFO
    'dwMaxBytesPerSec
    'dwFlags
    'dwCaps
    'dwStreams
    'dwSuggestedBufferSize
    'dwWidth As Integer
    'dwHeight As Integer
    'dwScale
    'dwRate
    'dwLength
    'dwEditCount
    'szFileType As String * 64
'End Type

Type BitmapInfo   'Varies
        bmiHeader As BITMAPINFOHEADER
        'bmiColors As String * 128 ' Array length is arbitrary; may be changed
End Type

Type BITMAPFILEHEADER   '14 Bytes
        bfType As Integer
        bfSize As Long
        bfReserved1 As Integer
```

```
        bfReserved2 As Integer
        bfOffBits As Long
End Type

Type PointAPI
    x As Integer
    Y As Integer
End Type

'  OpenFile() Structure
Type OFSTRUCT
        cBytes As String * 1
        fFixedDisk As String * 1
        nErrCode As Integer
        reserved As String * 4
        szPathName As String * 128
End Type

Type RECT
    Left As Integer
    Top As Integer
    Right As Integer
    Bottom As Integer
End Type

Type PaintStruct
    hDC As Integer
    fErase As Integer
    rcPaint As RECT
    fRestore As Integer
    fIncUpdate As Integer
    rgbReserved As Integer
End Type

Type FileInfo
   FileInfoFormat As String * 1          'Image Format
   FileInfoName As String * 13           'Name of Image
   FileInfoWidth As Integer              'Width
   FileInfoHeight As Integer             'Height
   FileInfoBitsPixel As Integer          'Resolution
   FileInfoSizeDisk As Long              'Bitmap size on disk
   FileInfoSizeMem As Long               'Bitmap size in memory
   FileInfoCompression As String * 20    'Compression type
   FileInfoViewPerspective As Integer    'View perspective
   FileInfoOrder As Integer              'Order RGB/BGR
End Type
Global FInfo As FileInfo

Type ChildData
    ValidBitmap As Integer
    Filename As String * 256
    FileInfo As FileInfo
    BitmapHandle As Integer
End Type

Type FixedFileInfo  '52 bytes
    Signature As Long
    StrucVersion As Long
    FileVersionMS As Long
    FileVersionLS As Long
    ProductVersionMS As Long
    ProductVersionLS As Long
    FileFlagsMask As Long
    FileFlags As Long
    FileOS As Long
    FileType As Long
    FileSubType As Long
    FileDateMS As Long
    FileDateLS As Long
End Type
```

```
Type HelpWinInfo  'size=12 + length of rgchMember
    wStructSize As Integer
    x As Integer
    Y As Integer
    dx As Integer
    dy As Integer
    wMax As Integer
    rgchMember As String * 16
End Type

Type TextMetric
    tmHeight As Integer
    tmAscent As Integer
    tmDescent As Integer
    tmInternalLeading As Integer
    tmExternalLeading As Integer
    tmAveCharWidth As Integer
    tmMaxCharWidth As Integer
    tmWeight As Integer
    tmItalic As String * 1
    tmUnderlined As String * 1
    tmStruckOut As String * 1
    tmFirstChar As String * 1
    tmLastChar As String * 1
    tmDefaultChar As String * 1
    tmBreakChar As String * 1
    tmPitchAndFamily As String * 1
    tmCharSet As String * 1
    tmOverhang As Integer
    tmDigitizedAspectX As Integer
    tmDigitizedAspectY As Integer
End Type


'DECLARATIONS SECTION -------------------------

Declare Function WinHelpByNum% Lib "USER" Alias "WinHelp" (ByVal hWnd%, ByVal lpHelpFile$, ByVal
 wCommand%, ByVal dwData&)
Declare Function WinHelp% Lib "USER" (ByVal hWnd%, ByVal lpHelpFile$, ByVal wCommand%, ByVal dwD
ata As Any)
Declare Function FindWindow% Lib "USER" (ByVal lpClassName As Any, ByVal lpWindowName As Any)
Declare Function FindWindowBystring% Lib "USER" Alias "FindWindow" (ByVal lpClassName$, ByVal lp
WindowName$)
Declare Function GetTopWindow% Lib "USER" (ByVal hWnd%)
Declare Function ChildWindowFromPointByNum% Lib "USER" Alias "ChildWindowFromPoint" (ByVal hWnd%
, ByVal Pnt&)
Declare Function GetWindow% Lib "USER" (ByVal hWnd%, ByVal wCmd%)
Declare Function GetWindowText Lib "USER" (ByVal hWnd As Integer, ByVal lpString As String, ByVa
l aint As Integer) As Integer
Declare Function GetDC% Lib "USER" (ByVal hWnd%)
Declare Function ReleaseDC% Lib "USER" (ByVal hWnd%, ByVal hDC%)
Declare Function GetFocus Lib "USER" () As Integer
Declare Function GetSystemMenu% Lib "USER" (ByVal hWnd%, ByVal bRevert%)
Declare Function ModifyMenu% Lib "USER" (ByVal hMenu%, ByVal nPosition%, ByVal wFlags%, ByVal wI
DNewItem%, ByVal lpString As Any)
Declare Function RemoveMenu% Lib "USER" (ByVal hMenu%, ByVal nPosition%, ByVal wFlags%)
Declare Sub PutFocus Lib "USER" Alias "SetFocus" (ByVal hWnd As Integer)
Declare Function SendMessage% Lib "USER" (ByVal hWnd%, ByVal wMsg%, ByVal wParam%, lparam As Any
)
Declare Function SendMessageByString% Lib "USER" Alias "SendMessage" (ByVal hWnd%, ByVal wMsg%,
ByVal wParam%, ByVal lparam$)
Declare Function SendMessageByNum% Lib "USER" Alias "SendMessage" (ByVal hWnd%, ByVal wMsg%, ByV
al wParam%, ByVal lparam&)
Declare Function GetSystemDirectory Lib "Kernel" (ByVal lpBuffer As String, ByVal nSize As Integ
er) As Integer
Declare Function GetPrivateProfileString% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyNam
e As Any, ByVal lpDefault$, ByVal lpReturnedString$, ByVal nSize%, ByVal lpFileName$)
Declare Function GetPrivateProfileInt% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyName$,
```

```
 ByVal nDefault%, ByVal lpFileName$)
Declare Function GetProfileString% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyName As An
y, ByVal lpDefault$, ByVal lpReturnedString$, ByVal nSize%)
Declare Function GetProfileInt% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyName$, ByVal
nDefault%)
Declare Function GetWindowsDirectory% Lib "Kernel" (ByVal lpBuffer$, ByVal nSize%)
Declare Function WritePrivateProfileString% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyN
ame As Any, ByVal lpString$, ByVal lpFileName$)
Declare Function WriteProfileString% Lib "Kernel" (ByVal lpApplicationName$, ByVal lpKeyName$, B
yVal lpString$)
Declare Function mciSendString Lib "MMSystem" (ByVal lpstrCommand As String, ByVal lpstrReturn A
s String, ByVal nSize As Integer, ByVal hCallback As Integer) As Long
Declare Function mciGetErrorString Lib "MMSystem" (ByVal dwError As Long, ByVal lpstrBuffer As S
tring, ByVal wLength As Integer) As Integer
Declare Function SSD2GetControlHwnd Lib "SSDATA2.VBX" (c As Control) As Integer
Declare Function GetFreeSpace& Lib "Kernel" (ByVal wFlags%)
Declare Function GetFreeSystemResources% Lib "USER" (ByVal fuSysRecorce%)
Declare Function GlobalAlloc Lib "Kernel" (ByVal wFlags As Integer, ByVal dwBytes As Long) As In
teger
Declare Function GlobalFree Lib "Kernel" (ByVal hMem As Integer) As Integer
Declare Function GlobalLock Lib "Kernel" (ByVal hMem As Integer) As Long
Declare Function GlobalUnlock Lib "Kernel" (ByVal hMem As Integer) As Integer
Declare Function OPenFile Lib "Kernel" (ByVal lpFileName As String, lpReOpenBuff As OFSTRUCT, By
Val wStyle As Integer) As Integer
Declare Function lclose Lib "Kernel" Alias "_lclose" (ByVal hFile As Integer) As Integer
Declare Function hwrite Lib "Kernel" Alias "_hwrite" (ByVal hFile As Integer, ByVal hpvBuffer As
 Long, ByVal wBytes As Long) As Long
Declare Function CreateSolidBrush Lib "GDI" (ByVal crColor As Long) As Integer
Declare Sub FillRect Lib "USER" (ByVal hDC As Integer, lpRect As RECT, ByVal hBrush As Integer)
Declare Function DeleteObject% Lib "GDI" (ByVal hObject%)
Declare Function GetBitmapBitsBynum& Lib "GDI" Alias "GetBitmapBits" (ByVal hBitmap%, ByVal dwCo
unt&, ByVal lpBits&)
Declare Function GetDialogBaseUnits Lib "USER" () As Long
Declare Function CreateWindow% Lib "USER" (ByVal lpClassName$, ByVal lpWindowName$, ByVal dwS
e&, ByVal x%, ByVal Y%, ByVal nWidth%, ByVal nHeight%, ByVal hwndParent%, ByVal hMenu%, ByVal I
nstance%, ByVal lpParam$)
Declare Function ShowWindow% Lib "USER" (ByVal hWnd%, ByVal nCmdShow%)
Declare Function DestroyWindow% Lib "USER" (ByVal hWnd%)
Declare Function SetWindowText% Lib "USER" (ByVal hWnd%, lpString$)
Declare Function BeginPaint% Lib "USER" (ByVal hWnd%, lpPaint As PaintStruct)
Declare Sub EndPaint Lib "USER" (ByVal hWnd%, lpPaint As PaintStruct)
Declare Function GetDeviceCaps% Lib "GDI" (ByVal hDC%, ByVal nIndex%)
Declare Function GetVersion Lib "Kernel" () As Long
Declare Function GetKeyboardType Lib "Keyboard" (ByVal nTypeFlag As Integer) As Integer
Declare Function GetWinFlags Lib "Kernel" () As Long
Declare Sub GetWindowRect Lib "USER" (ByVal hWnd As Integer, lpRect As RECT)
Declare Sub Rectangle Lib "GDI" (ByVal hDC As Integer, ByVal X1 As Integer, ByVal Y1 As Integer,
 ByVal X2 As Integer, ByVal Y2 As Integer)
Declare Function SelectObject% Lib "GDI" (ByVal hDC As Integer, ByVal hObject As Integer)
Declare Function GetTickCount& Lib "USER" ()
Declare Function GetFileVersionInfo% Lib "ver.dll" (ByVal lpszFileName$, ByVal Handle&, ByVal cb
Buf&, ByVal lpvData&)
Declare Function GetFileVersionInfoSize% Lib "ver.dll" (ByVal lpszFileName$, lpdwHandle&)
Declare Function Escape Lib "GDI" (ByVal hDC As Integer, ByVal nEscape As Integer, ByVal nCount
As Integer, lplnData As Any, lpOutData As Any) As Integer
Declare Function GetTextMetrics Lib "GDI" (ByVal hDC As Integer, lpMetrics As TextMetric) As Int
eger
Declare Function CreateCompatibleDC% Lib "GDI" (ByVal hDC%)
Declare Function StretchBlt% Lib "GDI" (ByVal hDC%, ByVal x%, ByVal Y%, ByVal nWidth%, ByVal nHe
ight%, ByVal hSrcDC%, ByVal XSrc%, ByVal YSrc%, ByVal nSrcWidth%, ByVal nSrcHeight%, ByVal dwRop
&)
Declare Function StretchDIBits% Lib "GDI" (ByVal hDC%, ByVal x%, ByVal Y%, ByVal nWidth%, ByVal
nHeight%, ByVal XSrc%, ByVal YSrc%, ByVal nSrcWidth%, ByVal nSrcHeight%, ByVal lpBits&, lpBi
fo As BitmapInfo, ByVal wUseage%, ByVal dwRop&)
Declare Function DeleteDC% Lib "GDI" (ByVal hDC%)
Declare Function DiskSpaceFree Lib "SETUPKIT.DLL" () As Long
Declare Function sndPlaySound% Lib "MMSYSTEM.DLL" (ByVal lpszSoundName$, ByVal wFlags%)
Declare Function LineTo% Lib "GDI" (ByVal hDC%, ByVal x%, ByVal Y%)
Declare Function MoveTo% Lib "GDI" (ByVal hDC%, ByVal x%, ByVal Y%)
Declare Function SetROP2% Lib "GDI" (ByVal hDC%, ByVal DrawMode%)
```

```
Declare Function GetStockObject% Lib "GDI" (ByVal fnObject%)
Declare Function CreatePen% Lib "GDI" (ByVal hPen%, ByVal nWidth%, ByVal nColor As Any)
Declare Sub SetWindowPos Lib "USER" (ByVal hWnd%, ByVal hWndInsertAfter%, ByVal x%, ByVal Y%, By
Val cX%, ByVal cY%, ByVal wFlags%)
Declare Function SetClipBoardData% Lib "USER" (ByVal mFormat%, ByVal hMem%)
Declare Function OpenClipboard% Lib "USER" (ByVal hWnd%)
Declare Function CloseClipboard% Lib "USER" ()
Declare Function SetVoiceAccent% Lib "Sound" (ByVal nVoice%, ByVal nTempo%, ByVal nVolume%, ByVa
l nMode%, ByVal nPitch%)
Declare Function SetVoiceEnvelope% Lib "Sound" (ByVal nVoice%, ByVal nShape%, ByVal nRepeat%)
Declare Function SetVoiceNote% Lib "Sound" (ByVal nVoice%, ByVal nValue%, ByVal nLength%, ByVal
nCdots%)
Declare Function SetVoiceQueueSize% Lib "Sound" (ByVal nVoice%, ByVal nBytes%)
Declare Function SetVoiceSound% Lib "Sound" (ByVal nVoice%, ByVal lFrequency&, ByVal nDuration%)
Declare Function SetVoiceThreshold% Lib "Sound" (ByVal nVoice%, ByVal nNotes%)
Declare Sub CloseSound Lib "Sound" ()
Declare Function OpenSound% Lib "Sound" ()
Declare Function StartSound% Lib "Sound" ()
Declare Function GetDriveType% Lib "Kernel" (ByVal nDrive%)
Declare Function BitBLt% Lib "GDI" (ByVal hDestDC%, ByVal x%, ByVal Y%, ByVal nWidth%, ByVal nHe
ight%, ByVal hSrcDC%, ByVal XSrc%, ByVal YSrc%, ByVal dwRop&)
'SpyWorks declarations ------------
Declare Function dwGetAddressForObject Lib "dwSpyDll.dll" (object As Any) As Long
Declare Function dwGetAddressForVBString Lib "dwSpyDll.dll" (Strg As String) As Long
Declare Sub dwCopyDataBynum Lib "dwSpyDll.dll" Alias "dwCopyData" (ByVal source&, ByVal dest&, B
yVal nCount%)
Declare Function dwPOINTAPItoLong& Lib "dwSpyDll.dll" (Pp As PointAPI)
Declare Function dwGetInstance% Lib "dwSpyDll.dll" ()
Declare Function dwGetControlHwnd% Lib "dwSpyDll.dll" (hctl As Control)
'Video for Windows declarations ----------
Declare Function videoGetNumDevs Lib "msvideo.dll" () As Long
Declare Function videoOpen Lib "msvideo.dll" (ByVal lphVideo&, ByVal dwDevice&, ByVal dwFlags&)
As Long
Declare Function videoClose Lib "msvideo.dll" (ByVal hVideo%) As Long
Declare Function videoConfigure Lib "msvideo.dll" (ByVal hVideo%, ByVal Msg%, ByVal dwFlags&, By
Val lpdwReturn&, ByVal lpData1&, ByVal dwSize1&, ByVal lpData2&, ByVal dwSize2&) As Long
Declare Function videoFrame Lib "msvideo.dll" (ByVal hVideo%, ByVal lpVHdr&) As Long
Declare Function DrawDibOpen Lib "msvideo.dll" () As Integer
Declare Function DrawDibBegin Lib "msvideo.dll" (ByVal hDD%, ByVal hwd%, ByVal bw%, ByVal bh%, B
yVal btmp&, ByVal bbw%, ByVal bbh%, ByVal dfl%) As Integer
Declare Function DrawDibDraw Lib "msvideo.dll" (ByVal hDD%, ByVal hwd%, ByVal xdist%, ByVal ydis
t%, ByVal bw%, ByVal bh%, ByVal btmp&, ByVal bmt As Any, ByVal XSrc%, ByVal YSrc%, ByVal bbw%, B
yVal bbh%, ByVal dfl%) As Integer
Declare Function DrawDibEnd Lib "msvideo.dll" (ByVal hDD%) As Integer
Declare Function DrawDibClose Lib "msvideo.dll" (ByVal hDD%) As Integer
Declare Function DrawDibSetPalette Lib "msvideo.dll" (ByVal hDD%, ByVal hpal%) As Integer
Declare Function DrawDibRealize Lib "msvideo.dll" (ByVal hDD%, ByVal hDC%, ByVal fBackground%) A
s Integer
Declare Function DrawDibGetPalette Lib "msvideo.dll" (ByVal hDD%) As Integer
Declare Function DrawDibChangePalette Lib "msvideo.dll" (ByVal hDD%, ByVal iStart%, ByVal iLen%,
 ByVal lppe%) As Integer
'Sheriden declarations --------
Declare Function SSD2GetControlHwnd Lib "SSDATA2.VBX" (c As Control) As Integer
'EDILzssa.dll call
Declare Function LZSSPackFile% Lib "edilzssa.dll" (ByVal SrcFile$, ByVal DestFile$)
Declare Function LZSSUnPackFile% Lib "edilzssa.dll" (ByVal SrcFile$, ByVal DestFile$)
'Pptr decs --------
Declare Function Pprtr Lib "PPRTR.DLL" (ByVal hWnd As Integer, ByVal ppSelection As Integer, ByV
al PPNewValue As Integer, ByVal ppAction As Integer) As Integer
Declare Function DefPrtr Lib "PPRTR.DLL" (ByVal newone As String, ByVal oldone As String) As Int
eger
Declare Function Prtrs Lib "PPRTR.DLL" (ByVal plist As String) As Integer
'Declare Function PrtrCap Lib "PPRTR.DLL" (ndc As DEVCAP) As Integer
Declare Function GetPrtr Lib "PPRTR.DLL" (ByVal DefPrtr As String) As Integer
Declare Function GetPort Lib "PPRTR.DLL" (ByVal PtrPort As String) As Integer

'GLOBAL CONSTANTS SECTION -----------------

Global Const GW_HWNDFIRST = 0
Global Const GW_HWNDNEXT = 2
```

```
Global Const DATA_ACTIONCANCEL = 0
Global Const DATA_ACTIONMOVEFIRST = 1
Global Const DATA_ACTIONMOVEPREVIOUS = 2
Global Const DATA_ACTIONMOVENEXT = 3
Global Const DATA_ACTIONMOVELAST = 4
Global Const DATA_ACTIONADDNEW = 5
Global Const DATA_ACTIONUPDATE = 6
Global Const DATA_ACTIONDELETE = 7
Global Const DATA_ACTIONFIND = 8
Global Const DATA_ACTIONBOOKMARK = 9
Global Const DATA_ACTIONCLOSE = 10
Global Const DATA_ACTIONUNLOAD = 11

' Clipboard formats
Global Const CF_LINK = &HBF00
Global Const CF_TEXT = 1
Global Const CF_BITMAP = 2
Global Const CF_METAFILE = 3
Global Const CF_DIB = 8
Global Const CF_PALETTE = 9

' DragOver
Global Const ENTER = 0
Global Const LEAVE = 1
Global Const OVER = 2

' Drag (controls)
Global Const CANCEL = 0
Global Const BEGIN_DRAG = 1
Global Const END_DRAG = 2

' Show parameters
Global Const MODAL = 1
Global Const MODELESS = 0

' Arrange Method for MDI Forms
Global Const CASCADE = 0
Global Const TILE_HORIZONTAL = 1
Global Const TILE_VERTICAL = 2
Global Const ARRANGE_ICONS = 3

'ZOrder Method
Global Const BRINGTOFRONT = 0
Global Const SENDTOBACK = 1

' Key Codes
Global Const KEY_LBUTTON = &H1
Global Const KEY_RBUTTON = &H2
Global Const KEY_CANCEL = &H3
Global Const KEY_MBUTTON = &H4     ' NOT contiguous with L & RBUTTON
Global Const KEY_BACK = &H8
Global Const KEY_TAB = &H9
Global Const KEY_CLEAR = &HC
Global Const KEY_RETURN = &HD
Global Const KEY_SHIFT = &H10
Global Const KEY_CONTROL = &H11
Global Const KEY_MENU = &H12
Global Const KEY_PAUSE = &H13
Global Const KEY_CAPITAL = &H14
Global Const KEY_ESCAPE = &H1B
Global Const KEY_SPACE = &H20
Global Const KEY_PRIOR = &H21
Global Const KEY_NEXT = &H22
Global Const KEY_END = &H23
Global Const KEY_HOME = &H24
Global Const KEY_LEFT = &H25
Global Const KEY_UP = &H26
Global Const KEY_RIGHT = &H27
Global Const KEY_DOWN = &H28
```

```
Global Const KEY_SELECT = &H29
Global Const KEY_PRINT = &H2A
Global Const KEY_EXECUTE = &H2B
Global Const KEY_SNAPSHOT = &H2C
Global Const KEY_INSERT = &H2D
Global Const KEY_DELETE = &H2E
Global Const KEY_HELP = &H2F

' KEY_A thru KEY_Z are the same as their ASCII equivalents: 'A' thru 'Z'
' KEY_0 thru KEY_9 are the same as their ASCII equivalents: '0' thru '9'

Global Const KEY_NUMPAD0 = &H60
Global Const KEY_NUMPAD1 = &H61
Global Const KEY_NUMPAD2 = &H62
Global Const KEY_NUMPAD3 = &H63
Global Const KEY_NUMPAD4 = &H64
Global Const KEY_NUMPAD5 = &H65
Global Const KEY_NUMPAD6 = &H66
Global Const KEY_NUMPAD7 = &H67
Global Const KEY_NUMPAD8 = &H68
Global Const KEY_NUMPAD9 = &H69
Global Const KEY_MULTIPLY = &H6A
Global Const KEY_ADD = &H6B
Global Const KEY_SEPARATOR = &H6C
Global Const KEY_SUBTRACT = &H6D
Global Const KEY_DECIMAL = &H6E
Global Const KEY_DIVIDE = &H6F
Global Const KEY_F1 = &H70
Global Const KEY_F2 = &H71
Global Const KEY_F3 = &H72
Global Const KEY_F4 = &H73
Global Const KEY_F5 = &H74
Global Const KEY_F6 = &H75
Global Const KEY_F7 = &H76
Global Const KEY_F8 = &H77
Global Const KEY_F9 = &H78
Global Const KEY_F10 = &H79
Global Const KEY_F11 = &H7A
Global Const KEY_F12 = &H7B
Global Const KEY_F13 = &H7C
Global Const KEY_F14 = &H7D
Global Const KEY_F15 = &H7E
Global Const KEY_F16 = &H7F

Global Const KEY_NUMLOCK = &H90

' ErrNum (LinkError)
Global Const WRONG_FORMAT = 1
Global Const DDE_SOURCE_CLOSED = 6
Global Const TOO_MANY_LINKS = 7
Global Const DATA_TRANSFER_FAILED = 8

' QueryUnload
Global Const FORM_CONTROLMENU = 0
Global Const FORM_CODE = 1
Global Const APP_WINDOWS = 2
Global Const APP_TASKMANAGER = 3
Global Const FORM_MDIFORM = 4

' Colors
Global Const BLACK = &H0&
Global Const RED = &HFF&
Global Const GREEN = &HFF00&
Global Const YELLOW = &HFFFF&
Global Const Blue = &HFF0000
Global Const MAGENTA = &HFF00FF
Global Const CYAN = &HFFFF00
Global Const WHITE = &HFFFFFF
Global Const GRAY = &HC0C0C0
```

```
' System Colors
Global Const SCROLL_BARS = &H80000000           ' Scroll-bars gray area.
Global Const DESKTOP = &H80000001               ' Desktop.
Global Const ACTIVE_TITLE_BAR = &H80000002      ' Active window caption.
Global Const INACTIVE_TITLE_BAR = &H80000003    ' Inactive window caption.
Global Const MENU_BAR = &H80000004              ' Menu background.
Global Const WINDOW_BACKGROUND = &H80000005     ' Window background.
Global Const WINDOW_FRAME = &H80000006          ' Window frame.
Global Const MENU_TEXT = &H80000007             ' Text in menus.
Global Const WINDOW_TEXT = &H80000008           ' Text in windows.
Global Const TITLE_BAR_TEXT = &H80000009        ' Text in caption, size box, scroll-bar arrow bo
x..
Global Const ACTIVE_BORDER = &H8000000A         ' Active window border.
Global Const INACTIVE_BORDER = &H8000000B       ' Inactive window border.
Global Const APPLICATION_WORKSPACE = &H8000000C ' Background color of multiple document interfac
e (MDI) applications.
Global Const HIGHLIGHT = &H8000000D             ' Items selected item in a control.
Global Const HIGHLIGHT_TEXT = &H8000000E        ' Text of item selected in a control.
Global Const BUTTON_FACE = &H8000000F           ' Face shading on command buttons.
Global Const BUTTON_SHADOW = &H80000010         ' Edge shading on command buttons.
Global Const GRAY_TEXT = &H80000011             ' Grayed (disabled) text.  This color is set to
0 if the current display driver does not support a solid gray color.
Global Const BUTTON_TEXT = &H80000012           ' Text on push buttons.

' Enumerated Types

' Align (picture box)
Global Const none = 0
Global Const ALIGN_TOP = 1
Global Const ALIGN_BOTTOM = 2

' Alignment
Global Const LEFT_JUSTIFY = 0  ' 0 - Left Justify
Global Const RIGHT_JUSTIFY = 1 ' 1 - Right Justify
Global Const CENTER = 2        ' 2 - Center

' BorderStyle (form)
'Global Const NONE = 0           ' 0 - None
Global Const FIXED_SINGLE = 1   ' 1 - Fixed Single
Global Const SIZABLE = 2        ' 2 - Sizable (Forms only)
Global Const FIXED_DOUBLE = 3   ' 3 - Fixed Double (Forms only)

' DragMode
Global Const MANUAL = 0     ' 0 - Manual
Global Const AUTOMATIC = 1 ' 1 - Automatic

' DrawMode
Global Const BLACKNESS = 1       ' 1 - Blackness
Global Const NOT_MERGE_PEN = 2   ' 2 - Not Merge Pen
Global Const MASK_NOT_PEN = 3    ' 3 - Mask Not Pen
Global Const NOT_COPY_PEN = 4    ' 4 - Not Copy Pen
Global Const MASK_PEN_NOT = 5    ' 5 - Mask Pen Not
Global Const INVERT = 6          ' 6 - Invert
Global Const XOR_PEN = 7         ' 7 - Xor Pen
Global Const NOT_MASK_PEN = 8    ' 8 - Not Mask Pen
Global Const MASK_PEN = 9        ' 9 - Mask Pen
Global Const NOT_XOR_PEN = 10    ' 10 - Not Xor Pen
Global Const NOP = 11            ' 11 - Nop
Global Const MERGE_NOT_PEN = 12  ' 12 - Merge Not Pen
Global Const COPY_PEN = 13       ' 13 - Copy Pen
Global Const MERGE_PEN_NOT = 14  ' 14 - Merge Pen Not
Global Const MERGE_PEN = 15      ' 15 - Merge Pen
Global Const WHITENESS = 16      ' 16 - Whiteness

' DrawStyle
Global Const SOLID = 0         ' 0 - Solid
Global Const DASH = 1          ' 1 - Dash
Global Const DOT = 2           ' 2 - Dot
Global Const DASH_DOT = 3      ' 3 - Dash-Dot
Global Const DASH_DOT_DOT = 4  ' 4 - Dash-Dot-Dot
```

```
Global Const INVISIBLE = 5     ' 5 - Invisible
Global Const INVERSE = 6 ' 6 - Inverse

' FillStyle
' Global Const SOLID = 0            ' 0 - Solid
Global Const TRANSPARENT = 1        ' 1 - Transparent
Global Const HORIZONTAL_LINE = 2    ' 2 - Horizontal Line
Global Const VERTICAL_LINE = 3      ' 3 - Vertical Line
Global Const UPWARD_DIAGONAL = 4    ' 4 - Upward Diagonal
Global Const DOWNWARD_DIAGONAL = 5 ' 5 - Downward Diagonal
Global Const CROSS = 6              ' 6 - Cross
Global Const DIAGONAL_CROSS = 7     ' 7 - Diagonal Cross

' LinkMode (forms and controls)
' Global Const NONE = 0          ' 0 - None
Global Const LINK_SOURCE = 1     ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2     ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3     ' 3 - Notify (controls only)

' LinkMode (kept for VB1.0 compatibility, use new constants instead)
Global Const HOT = 1     ' 1 - Hot (controls only)
Global Const SERVER = 1 ' 1 - Server (forms only)
Global Const COLD = 2    ' 2 - Cold (controls only)


' ScaleMode
Global Const USER = 0         ' 0 - User
Global Const TWIPS = 1        ' 1 - Twip
Global Const POINTS = 2       ' 2 - Point
Global Const PIXELS = 3       ' 3 - Pixel
Global Const CHARACTERS = 4   ' 4 - Character
Global Const INCHES = 5       ' 5 - Inch
Global Const MILLIMETERS = 6 ' 6 - Millimeter
Global Const CENTIMETERS = 7 ' 7 - Centimeter

' ScrollBar
' Global Const NONE      = 0 ' 0 - None
Global Const HORIZONTAL = 1 ' 1 - Horizontal
Global Const VERTICAL = 2    ' 2 - Vertical
Global Const BOTH = 3        ' 3 - Both

' Shape
Global Const SHAPE_RECTANGLE = 0
Global Const SHAPE_SQUARE = 1
Global Const SHAPE_OVAL = 2
Global Const SHAPE_CIRCLE = 3
Global Const SHAPE_ROUNDED_RECTANGLE = 4
Global Const SHAPE_ROUNDED_SQUARE = 5

' WindowState
Global Const NORMAL = 0     ' 0 - Normal
Global Const MINIMIZED = 1 ' 1 - Minimized
Global Const MAXIMIZED = 2 ' 2 - Maximized

' Check Value
Global Const UNCHECKED = 0 ' 0 - Unchecked
Global Const Checked = 1    ' 1 - Checked
Global Const GRAYED = 2     ' 2 - Grayed

' Shift parameter masks
Global Const SHIFT_MASK = 1
Global Const CTRL_MASK = 2
Global Const ALT_MASK = 4

' Button parameter masks
Global Const LEFT_BUTTON = 1
Global Const RIGHT_BUTTON = 2
Global Const MIDDLE_BUTTON = 4
```

```
' SetAttr, Dir, GetAttr functions
Global Const ATTR_NORMAL = 0
Global Const ATTR_READONLY = 1
Global Const ATTR_HIDDEN = 2
Global Const ATTR_SYSTEM = 4
Global Const ATTR_VOLUME = 8
Global Const ATTR_DIRECTORY = 16
Global Const ATTR_ARCHIVE = 32

'Grid
'ColAlignment,FixedAlignment Properties
Global Const GRID_ALIGNLEFT = 0
Global Const GRID_ALIGNRIGHT = 1
Global Const GRID_ALIGNCENTER = 2

'Fillstyle Property
Global Const GRID_SINGLE = 0
Global Const GRID_REPEAT = 1

'-----------------------------------
'Common Dialog Control
'-----------------------------------
'Action Property
Global Const DLG_FILE_OPEN = 1
Global Const DLG_FILE_SAVE = 2
Global Const DLG_COLOR = 3
Global Const DLG_FONT = 4
Global Const DLG_PRINT = 5
Global Const DLG_HELP = 6

'File Open/Save Dialog Flags
Global Const OFN_READONLY = &H1&
Global Const OFN_OVERWRITEPROMPT = &H2&
Global Const OFN_HIDEREADONLY = &H4&
Global Const OFN_NOCHANGEDIR = &H8&
Global Const OFN_SHOWHELP = &H10&
Global Const OFN_NOVALIDATE = &H100&
Global Const OFN_ALLOWMULTISELECT = &H200&
Global Const OFN_EXTENTIONDIFFERENT = &H400&
Global Const OFN_PATHMUSTEXIST = &H800&
Global Const OFN_FILEMUSTEXIST = &H1000&
Global Const OFN_CREATEPROMPT = &H2000&
Global Const OFN_SHAREAWARE = &H4000&
Global Const OFN_NOREADONLYRETURN = &H8000&

'Color Dialog Flags
Global Const CC_RGBINIT = &H1&
Global Const CC_FULLOPEN = &H2&
Global Const CC_PREVENTFULLOPEN = &H4&
Global Const CC_SHOWHELP = &H8&

'Fonts Dialog Flags
Global Const CF_SCREENFONTS = &H1&
Global Const CF_PRINTERFONTS = &H2&
Global Const CF_BOTH = &H3&
Global Const CF_SHOWHELP = &H4&
Global Const CF_INITTOLOGFONTSTRUCT = &H40&
Global Const CF_USESTYLE = &H80&
Global Const CF_EFFECTS = &H100&
Global Const CF_APPLY = &H200&
Global Const CF_ANSIONLY = &H400&
Global Const CF_NOVECTORFONTS = &H800&
Global Const CF_NOSIMULATIONS = &H1000&
Global Const CF_LIMITSIZE = &H2000&
Global Const CF_FIXEDPITCHONLY = &H4000&
Global Const CF_WYSIWYG = &H8000&                    'must also have CF_SCREENFONTS & CF_PRI
NTERFONTS
Global Const CF_FORCEFONTEXIST = &H10000
Global Const CF_SCALABLEONLY = &H20000
Global Const CF_TTONLY = &H40000
```

```
Global Const CF_NOFACESEL = &H80000
Global Const CF_NOSTYLESEL = &H100000
Global Const CF_NOSIZESEL = &H200000

'Help Constants
Global Const HELP_CONTEXT = &H1           'Display topic in ulTopic
Global Const HELP_QUIT = &H2              'Terminate help
Global Const HELP_INDEX = &H3             'Display index
Global Const HELP_CONTENTS = &H3
Global Const HELP_HELPONHELP = &H4        'Display help on using help
Global Const HELP_SETINDEX = &H5          'Set the current Index for multi index help
Global Const HELP_SETCONTENTS = &H5
Global Const HELP_CONTEXTPOPUP = &H8
Global Const HELP_FORCEFILE = &H9
Global Const HELP_KEY = &H101             'Display topic for keyword in offabData
Global Const HELP_COMMAND = &H102
Global Const HELP_PARTIALKEY = &H105      'call the search engine in winhelp
Global Const HELP_SETWINPOS = &H203

'SpinOrientation
Global Const SPIN_VERTICAL = 0
Global Const SPIN_HORIZONTAL = 1

'  OpenFile() Flags
Global Const OF_READ = &H0
Global Const OF_WRITE = &H1
Global Const OF_READWRITE = &H2
Global Const OF_SHARE_COMPAT = &H0
Global Const OF_SHARE_EXCLUSIVE = &H10
Global Const OF_SHARE_DENY_WRITE = &H20
Global Const OF_SHARE_DENY_READ = &H30
Global Const OF_SHARE_DENY_NONE = &H40
Global Const OF_PARSE = &H100
Global Const OF_DELETE = &H200
Global Const OF_VERIFY = &H400
Global Const OF_CANCEL = &H800
Global Const OF_CREATE = &H1000
Global Const OF_PROMPT = &H2000
Global Const OF_EXIST = &H4000
Global Const OF_REOPEN = &H8000
Global Const OF_SEARCH = &H400    '???????

Global Const READAPI = 0          '  Flags for _lopen
Global Const WRITEAPI = 1
Global Const READ_WRITE = 2

'Global Memory Flags
Global Const GMEM_FIXED = &H0
Global Const GMEM_MOVEABLE = &H2
Global Const GMEM_NOCOMPACT = &H10
Global Const GMEM_NODISCARD = &H20
Global Const GMEM_ZEROINIT = &H40
Global Const GMEM_MODIFY = &H80
Global Const GMEM_DISCARDABLE = &H100
Global Const GMEM_NOT_BANKED = &H1000
Global Const GMEM_SHARE = &H2000
Global Const GMEM_DDESHARE = &H2000
Global Const GMEM_NOTIFY = &H4000
Global Const GMEM_LOWER = GMEM_NOT_BANKED

Global Const GHND = (GMEM_MOVEABLE Or GMEM_ZEROINIT)
Global Const GPTR = (GMEM_FIXED Or GMEM_ZEROINIT)

'Flags returned by GlobalFlags (in addition to GMEM_DISCARDABLE)
Global Const GMEM_DISCARDED = &H4000
Global Const GMEM_LOCKCOUNT = &HFF

'Used in list tabbing
Global Const WM_USER = &H400
Global Const LB_RESETCONTENT = WM_USER + 5
```

```
Global Const LB_SETTABSTOPS = WM_USER + 19
Global Const LB_FINDSTRING = WM_USER + 16
Global Const LB_FINDSTRINGEXACT = WM_USER + 35
Global Const LB_SELECTSTRING = WM_USER + 13
Global Const LB_SELITEMRANGE = WM_USER + 28

'Other message constants
Global Const EM_UNDO = WM_USER + 23
Global Const EM_CANUNDO = WM_USER + 22
Global Const WM_UNDO = &H304
Global Const WM_CUT = &H300
Global Const WM_COPY = &H301
Global Const WM_PASTE = &H302
Global Const WM_CLEAR = &H303
Global Const WM_MENUSELECT = &H11F
Global Const WM_QUIT = &H12
Global Const WM_CLOSE = &H10

Global Const MF_BYCOMMAND = &H0
Global Const MF_BYPOSITION = &H400
Global Const MF_BITMAP = &H4
Global Const MF_DISABLED = &H2
Global Const MF_GRAYED = &H1

Global Const SC_RESTORE = &HF120
Global Const SC_MOVE = &HF010
Global Const SC_SIZE = &HF000
Global Const SC_MINIMIZE = &HF020
Global Const SC_MAXIMIZE = &HF030
Global Const SC_CLOSE = &HF060
Global Const SC_TASKLIST = &HF130

Global Const SW_HIDE = 0
Global Const SW_SHOWNORMAL = 1
Global Const SW_NORMAL = 1
Global Const SW_SHOWMINIMIZED = 2
Global Const SW_SHOWMAXIMIZED = 3
Global Const SW_MAXIMIZE = 3
Global Const SW_SHOWNOACTIVATE = 4
Global Const SW_SHOW = 5
Global Const SW_MINIMIZE = 6
Global Const SW_SHOWMINNOACTIVE = 7
Global Const SW_SHOWNA = 8
Global Const SW_RESTORE = 9

Global Const WS_OVERLAPPED = &H0&
Global Const WS_POPUP = &H80000000
Global Const WS_CHILD = &H40000000
Global Const WS_MINIMIZE = &H20000000
Global Const WS_VISIBLE = &H10000000
Global Const WS_DISABLED = &H8000000
Global Const WS_CLIPSIBLINGS = &H4000000
Global Const WS_CLIPCHILDREN = &H2000000
Global Const WS_MAXIMIZE = &H1000000
Global Const WS_CAPTION = &HC00000   '  WS_BORDER Or WS_DLGFRAME
Global Const WS_BORDER = &H800000
Global Const WS_DLGFRAME = &H400000
Global Const WS_VSCROLL = &H200000
Global Const WS_HSCROLL = &H100000
Global Const WS_SYSMENU = &H80000
Global Const WS_THICKFRAME = &H40000
Global Const WS_GROUP = &H20000
Global Const WS_TABSTOP = &H10000

Global Const WS_MINIMIZEBOX = &H20000
Global Const WS_MAXIMIZEBOX = &H10000

Global Const WS_TILED = WS_OVERLAPPED
Global Const WS_ICONIC = WS_MINIMIZE
Global Const WS_SIZEBOX = WS_THICKFRAME
```

```
'  Common Window Styles
Global Const WS_OVERLAPPEDWINDOW = (WS_OVERLAPPED Or WS_CAPTION Or WS_SYSMENU Or WS_THICKFRAME O
r WS_MINIMIZEBOX Or WS_MAXIMIZEBOX)
Global Const WS_POPUPWINDOW = (WS_POPUP Or WS_BORDER Or WS_SYSMENU)
Global Const WS_CHILDWINDOW = (WS_CHILD)
Global Const WS_TILEDWINDOW = (WS_OVERLAPPEDWINDOW)

'  Extended Window Styles
Global Const WS_EX_DLGMODALFRAME = &H1&
Global Const WS_EX_NOPARENTNOTIFY = &H4&
Global Const WS_EX_TOPMOST = &H8&
Global Const WS_EX_ACCEPTFILES = &H10&
Global Const WS_EX_TRANSPARENT = &H20&

''  GetDeviceCaps() Device Parameters
Global Const DRIVERVERSION = 0  '  Device driver version
Global Const TECHNOLOGY = 2 '  Device classification
Global Const HORZSIZE = 4    '  Horizontal size in millimeters
Global Const VERTSIZE = 6    '  Vertical size in millimeters
Global Const HORZRES = 8     '  Horizontal width in pixels
Global Const VERTRES = 10    '  Vertical width in pixels
Global Const BITSPIXEL = 12 '  Number of bits per pixel
Global Const PLANES = 14     '  Number of planes
Global Const NUMBRUSHES = 16     '  Number of brushes the device has
Global Const NUMPENS = 18    '  Number of pens the device has
Global Const NUMMARKERS = 20     '  Number of markers the device has
Global Const NUMFONTS = 22   '  Number of fonts the device has
Global Const NUMCOLORS = 24 '  Number of colors the device supports
Global Const PDEVICESIZE = 26    '  Size required for device descriptor
Global Const CURVECAPS = 28 '  Curve capabilities
Global Const LINECAPS = 30   '  Line capabilities
Global Const POLYGONALCAPS = 32 '  Polygonal capabilities
Global Const TEXTCAPS = 34   '  Text capabilities
Global Const CLIPCAPS = 36   '  Clipping capabilities
Global Const RASTERCAPS = 38     '  Bitblt capabilities
Global Const ASPECTX = 40    '  Length of the X leg
Global Const ASPECTY = 42    '  Length of the Y leg
Global Const ASPECTXY = 44   '  Length of the hypotenuse

Global Const LOGPIXELSX = 88     '  Logical pixels/inch in X
Global Const LOGPIXELSY = 90     '  Logical pixels/inch in Y

Global Const SIZEPALETTE = 104   '  Number of entries in physical palette
Global Const NUMRESERVED = 106   '  Number of reserved entries in palette
Global Const COLORRES = 108 '  Actual color resolution

'' Version control flags
Global Const VS_FFI_SIGNATURE = &HFEEF04BD
Global Const VS_FFI_STRUCVERSION = &H10000
Global Const VS_FFI_FILEFLAGSMASK = &H3F&
Global Const VS_FF_DEBUG = &H1&
Global Const VS_FF_PRERELEASE = &H2&
Global Const VS_FF_PATCHED = &H4&
Global Const VS_FF_PRIVATEBUILD = &H8&
Global Const VS_FF_INFOINFERRED = &H10&
Global Const VS_FF_SPECIALBUILD = &H20&

'' Version control OS flags
Global Const VOS_UNKNOWN = &H0&
Global Const VOS_DOS = &H10000
Global Const VOS_OS216 = &H20000
Global Const VOS_OS232 = &H30000
Global Const VOS_NT = &H40000
Global Const VOS__BASE = &H0&
Global Const VOS__WINDOWS16 = &H1&
Global Const VOS__PM16 = &H2&
Global Const VOS__PM32 = &H3&
Global Const VOS__WINDOWS32 = &H4&
Global Const VOS_DOS_WINDOWS16 = &H10001
```

```
Global Const VOS_DOS_WINDOWS32 = &H10004
Global Const VOS_OS216_PM16 = &H20002
Global Const VOS_OS232_PM32 = &H30003
Global Const VOS_NT_WINDOWS32 = &H40004

'' Version control file types
Global Const VFT_UNKNOWN = &H0&
Global Const VFT_APP = &H1&
Global Const VFT_DLL = &H2&
Global Const VFT_DRV = &H3&
Global Const VFT_FONT = &H4&
Global Const VFT_VXD = &H5&
Global Const VFT_STATIC_LIB = &H7&

' VS_VERSION.dwFileSubtype for VFT_WINDOWS_DRV
Global Const VFT2_UNKNOWN = &H0&
Global Const VFT2_DRV_PRINTER = &H1&
Global Const VFT2_DRV_KEYBOARD = &H2&
Global Const VFT2_DRV_LANGUAGE = &H3&
Global Const VFT2_DRV_DISPLAY = &H4&
Global Const VFT2_DRV_MOUSE = &H5&
Global Const VFT2_DRV_NETWORK = &H6&
Global Const VFT2_DRV_SYSTEM = &H7&
Global Const VFT2_DRV_INSTALLABLE = &H8&
Global Const VFT2_DRV_SOUND = &H9&
Global Const VFT2_DRV_COMM = &HA&

' VS_VERSION.dwFileSubtype for VFT_WINDOWS_FONT
Global Const VFT2_FONT_RASTER = &H1&
Global Const VFT2_FONT_VECTOR = &H2&
Global Const VFT2_FONT_TRUETYPE = &H3&

'' VerFindFile() flags
Global Const VFFF_ISSHAREDFILE = &H1

Global Const VFF_CURNEDEST = &H1
Global Const VFF_FILEINUSE = &H2
Global Const VFF_BUFFTOOSMALL = &H4

'' VerInstallFile() flags
Global Const VIFF_FORCEINSTALL = &H1
Global Const VIFF_DONTDELETEOLD = &H2

Global Const VIF_TEMPFILE = &H1&
Global Const VIF_MISMATCH = &H2&
Global Const VIF_SRCOLD = &H4&

Global Const VIF_DIFFLANG = &H8&
Global Const VIF_DIFFCODEPG = &H10&
Global Const VIF_DIFFTYPE = &H20&

Global Const VIF_WRITEPROT = &H40&
Global Const VIF_FILEINUSE = &H80&
Global Const VIF_OUTOFSPACE = &H100&
Global Const VIF_ACCESSVIOLATION = &H200&
Global Const VIF_SHARINGVIOLATION = &H400&
Global Const VIF_CANNOTCREATE = &H800&
Global Const VIF_CANNOTDELETE = &H1000&
Global Const VIF_CANNOTRENAME = &H2000&
Global Const VIF_CANNOTDELETECUR = &H4000&
Global Const VIF_OUTOFMEMORY = &H8000&

Global Const VIF_CANNOTREADSRC = &H10000
Global Const VIF_CANNOTREADDST = &H20000

Global Const VIF_BUFFTOOSMALL = &H40000

'Escape function constants
Global Const GETPAGESIZE = 12
Global Const GETPAGEORIENT = 30
```

```
'sndPlaySound constants
Global Const SND_SYNC = &H0
Global Const SND_ASYNC = &H1
Global Const SND_NODEFAULT = &H2
Global Const SND_LOOP = &H8
Global Const SND_NOSTOP = &H10

'Float window constants
Global Const SWP_NOMOVE = 2
Global Const SWP_NOSIZE = 1
Global Const HWND_TOP = 0
Global Const HWND_BOTTOM = 1
Global Const HWND_TOPMOST = -1
Global Const HWND_NOTOPMOST = -2
Global Const FLOAT = 1, SINK = 0

'used in GetDriveType
Global Const DRIVE_REMOVABLE = 2
Global Const DRIVE_FIXED = 3
Global Const DRIVE_REMOTE = 4

' Option argument values (CreateDynaset, etc)
Global Const DB_DENYWRITE = &H1
Global Const DB_DENYREAD = &H2
Global Const DB_READONLY = &H4
Global Const DB_APPENDONLY = &H8
Global Const DB_INCONSISTENT = &H10
Global Const DB_CONSISTENT = &H20
Global Const DB_SQLPASSTHROUGH = &H40

' SetDataAccessOption
Global Const DB_OPTIONINIPATH = 1

' Field Attributes
Global Const DB_FIXEDFIELD = &H1
Global Const DB_VARIABLEFIELD = &H2
Global Const DB_AUTOINCRFIELD = &H10
Global Const DB_UPDATABLEFIELD = &H20

' Field Data Types
Global Const DB_BOOLEAN = 1
Global Const DB_BYTE = 2
Global Const DB_INTEGER = 3
Global Const DB_LONG = 4
Global Const DB_CURRENCY = 5
Global Const DB_SINGLE = 6
Global Const DB_DOUBLE = 7
Global Const DB_DATE = 8
Global Const DB_TEXT = 10
Global Const DB_LONGBINARY = 11
Global Const DB_MEMO = 12

' TableDef Attributes
Global Const DB_ATTACHEXCLUSIVE = &H10000
Global Const DB_ATTACHSAVEPWD = &H20000
Global Const DB_SYSTEMOBJECT = &H80000002
Global Const DB_ATTACHEDTABLE = &H40000000
Global Const DB_ATTACHEDODBC = &H20000000

' ListTables TableType
Global Const DB_TABLE = 1
Global Const DB_QUERYDEF = 5

' ListTables Attributes (for QueryDefs)
Global Const DB_QACTION = &HF0
Global Const DB_QCROSSTAB = &H10
Global Const DB_QDELETE = &H20
Global Const DB_QUPDATE = &H30
Global Const DB_QAPPEND = &H40
```

```
Global Const DB_QMAKETABLE = &H50

' ListIndexes IndexAttributes values
Global Const DB_UNIQUE = 1
Global Const DB_PRIMARY = 2
Global Const DB_PROHIBITNULL = 4
Global Const DB_IGNORENULL = 8
' ListIndexes FieldAttributes value
Global Const DB_DESCENDING = 1  'For each field in Index

' CreateDatabase and CompactDatabase Language constants
Global Const DB_LANG_GENERAL = ";LANGID=0x0809;CP=1252;COUNTRY=0"

' CreateDatabase and CompactDatabase options
Global Const DB_VERSION10 = 1        ' Microsoft Access Version 1.0
Global Const DB_ENCRYPT = 2          ' Make database encrypted.
Global Const DB_DECRYPT = 4          ' Decrypt database while compacting.

'Collating order values
Global Const DB_SORTGENERAL = 256    ' Sort by EFGPI rules (English, French, German,Portuguese,
Italian)
Global Const DB_SORTUNDEFINED = -1   ' Sort rules are undefined or unknown

'BitBlt constants
Const SRCCOPY = &HCC0020

'Pptr constants
Global Const DMORIENT_PORTRAIT = 1
Global Const DMORIENT_LANDSCAPE = 2
Global Const DM_ORIENTATION = &H1
Global Const DM_PAPERSIZE = &H2
Global Const DM_PAPERLENGTH = &H4
Global Const DM_PAPERWIDTH = &H8
Global Const DM_SCALE = &H10
Global Const DM_COPIES = &H100
Global Const DM_DEFAULTSOURCE = &H200
Global Const DM_PRINTQUALITY = &H400
Global Const DM_COLOR = &H800
Global Const DM_DUPLEX = &H1000
Global Const DM_YRESOLUTION = &H2000
Global Const DM_TTOPTION = &H4000

'MCI Messages
Global Const MCIWNDM_GETDEVICEID = WM_USER + 100
Global Const MCIWNDM_SENDSTRING = WM_USER + 101
Global Const MCIWNDM_GETPOSITION = WM_USER + 102
Global Const MCIWNDM_GETSTART = WM_USER + 103
Global Const MCIWNDM_GETLENGTH = WM_USER + 104
Global Const MCIWNDM_GETEND = WM_USER + 105
Global Const MCIWNDM_GETMODE = WM_USER + 106
Global Const MCIWNDM_EJECT = WM_USER + 107
Global Const MCIWNDM_SETZOOM = WM_USER + 108
Global Const MCIWNDM_GETZOOM = WM_USER + 109
Global Const MCIWNDM_SETVOLUME = WM_USER + 110
Global Const MCIWNDM_GETVOLUME = WM_USER + 111
Global Const MCIWNDM_SETSPEED = WM_USER + 112
Global Const MCIWNDM_GETSPEED = WM_USER + 113
Global Const MCIWNDM_SETREPEAT = WM_USER + 114
Global Const MCIWNDM_GETREPEAT = WM_USER + 115
Global Const MCIWNDM_REALIZE = WM_USER + 118
Global Const MCIWNDM_SETTIMEFORMAT = WM_USER + 119
Global Const MCIWNDM_GETTIMEFORMAT = WM_USER + 120
Global Const MCIWNDM_VALIDATEMEDIA = WM_USER + 121
Global Const MCIWNDM_PLAYFROM = WM_USER + 122
Global Const MCIWNDM_PLAYTO = WM_USER + 123
Global Const MCIWNDM_GETFILENAME = WM_USER + 124
Global Const MCIWNDM_GETDEVICE = WM_USER + 125
Global Const MCIWNDM_GETPALETTE = WM_USER + 126
Global Const MCIWNDM_SETPALETTE = WM_USER + 127
Global Const MCIWNDM_GETERROR = WM_USER + 128
```

```
Global Const MCIWNDM_SETTIMERS = WM_USER + 129
Global Const MCIWNDM_SETACTIVETIMER = WM_USER + 130
Global Const MCIWNDM_SETINACTIVETIMER = WM_USER + 131
Global Const MCIWNDM_GETACTIVETIMER = WM_USER + 132
Global Const MCIWNDM_GETINACTIVETIMER = WM_USER + 133
Global Const MCIWNDM_NEW = WM_USER + 134
Global Const MCIWNDM_CHANGESTYLES = WM_USER + 135
Global Const MCIWNDM_GETSTYLES = WM_USER + 136
Global Const MCIWNDM_GETALIAS = WM_USER + 137
Global Const MCIWNDM_RETURNSTRING = WM_USER + 138
Global Const MCIWNDM_PLAYREVERSE = WM_USER + 139
Global Const MCIWNDM_GET_SOURCE = WM_USER + 140
Global Const MCIWNDM_PUT_SOURCE = WM_USER + 141
Global Const MCIWNDM_GET_DEST = WM_USER + 142
Global Const MCIWNDM_PUT_DEST = WM_USER + 143
Global Const MCIWNDM_CAN_PLAY = WM_USER + 144
Global Const MCIWNDM_CAN_WINDOW = WM_USER + 145
Global Const MCIWNDM_CAN_RECORD = WM_USER + 146
Global Const MCIWNDM_CAN_SAVE = WM_USER + 147
Global Const MCIWNDM_CAN_EJECT = WM_USER + 148
Global Const MCIWNDM_CAN_CONFIG = WM_USER + 149
Global Const MCIWNDM_PALETTEKICK = WM_USER + 150
Global Const MCIWNDM_OPENINTERFACE = WM_USER + 151


Sub AddGraphSeconds()
    'find out how many seconds
    TblVar.MoveLast
    NumSecs = TblVar.Fields("SECONDS") + 1
    'NumSecs = AVILength / 1000
    IntSecs = NumSecs / 10
    TblVar.MoveFirst
    'move and set caption for each label
    GrLeft% = f_data.pnlGraph.Left
    FontTmp$ = f_data.FontName
    FSTmp = f_data.FontSize
    f_data.FontName = "Small Fonts"
    f_data.FontSize = 6.75
    For x% = 0 To 9
        T% = Int((x% + 1) * IntSecs)
        S$ = Format$(Int(T% / 60), "00") & ":" & Format$(T% Mod 60, "00")
        f_data.GSecs(x%).Move Int(GrLeft% + (x% + 1) * f_data.pnlGraph.Width / 10 - f_data.TextW
idth(S$) / 2)
        f_data.GSecs(x%).Caption = S$
    Next x%
    f_data.FontName = FontTmp$
    f_data.FontSize = FSTmp
End Sub

Sub centerform(F As Form)
    F.Left = (screen.Width - F.Width) / 2
    F.Top = (screen.Height - F.Height) / 2
End Sub

Function CleanString(SS As String) As String
    c$ = ""
    x = 1
    Do While x <= Len(SS)
        T$ = Mid$(SS, x, 1)
        Select Case T$
            Case "&"
                c$ = c$ & "&&"
            Case Else
                c$ = c$ & T$
        End Select
        x = x + 1
    Loop
    CleanString = c$
End Function
```

```
Sub ClearCurrentDB()
    'clear list of variables and query text
    f_query.List1.Clear
    f_query.txtQuery = ""
    'clear data graph
    f_data.pnlGraph.Cls
    f_data.MaxPointer.Visible = False
    f_data.MaxPointer2.Visible = False
    f_data.MaxPointer3.Visible = False
    f_data.MinPointer.Visible = False
    f_data.MinPointer2.Visible = False
    f_data.MinPointer3.Visible = False
    f_data.LeadinPointer.Visible = False
    f_data.LeadoutPointer.Visible = False
    f_data.MaxPointer.Left = 375
    f_data.MaxPointer2.Left = 375
    f_data.MaxPointer3.Left = 375
    f_data.MinPointer.Left = 375
    f_data.MinPointer2.Left = 375
    f_data.MinPointer3.Left = 375
    f_data.LeadinPointer.Left = 375
    f_data.LeadoutPointer.Left = 375
    f_data.pnlSlider.Left = 435
    f_data.Label2.Caption = "10"
    f_data.Label3.Caption = "0"
End Sub

Sub CreateNewModDB(FN$)
    Dim Td1 As New tabledef
    Dim I1 As New Index, I2 As New Index, I3 As New Index
    Dim I4 As New Index, I5 As New Index, I6 As New Index
    Dim I7 As New Index, I8 As New Index, I9 As New Index
    Dim I10 As New Index, I11 As New Index, I12 As New Index
    Dim I13 As New Index, I14 As New Index, I15 As New Index
    Dim I16 As New Index, I17 As New Index, I18 As New Index
    Dim I19 As New Index, I20 As New Index, I21 As New Index
    Dim I22 As New Index
    Dim F1 As New field, F2 As New field, F3 As New field
    Dim F4 As New field, F5 As New field, F6 As New field
    Dim F7 As New field, F8 As New field, F9 As New field
    Dim F10 As New field, F11 As New field, F12 As New field
    Dim F13 As New field, F14 As New field, F15 As New field
    Dim F16 As New field, F17 As New field, F18 As New field
    Dim F19 As New field, F20 As New field, F21 As New field
    Dim F22 As New field

    Err = 0

    If Exists(FN$) > 0 Then Kill FN$
    Set NewDB = CreateDatabase(FN$, DB_LANG_GENERAL)
    If NewDB Is Nothing Then MsgBox "Couldn't create database " & Client: Exit Sub
    If Err <> 0 Then MsgBox "Error creating database: " & Error$(Err)
    'On Error GoTo EH_CreateNewDB

    '------ Data Table
    Td1.Name = "T_DATA"
    F1.Name = "SECONDS": F1.Type = DB_INTEGER: Td1.Fields.Append F1
    If VArr(0) <> "" Then F2.Name = XArr(0): F2.Type = DB_SINGLE: Td1.Fields.Append F2
    If VCtr > 1 Then F3.Name = XArr(1): F3.Type = DB_SINGLE: Td1.Fields.Append F3
    If VCtr > 2 Then F4.Name = XArr(2): F4.Type = DB_SINGLE: Td1.Fields.Append F4
    If VCtr > 3 Then F5.Name = XArr(3): F5.Type = DB_SINGLE: Td1.Fields.Append F5
    If VCtr > 4 Then F6.Name = XArr(4): F6.Type = DB_SINGLE: Td1.Fields.Append F6
    If VCtr > 5 Then F7.Name = XArr(5): F7.Type = DB_SINGLE: Td1.Fields.Append F7
    If VCtr > 6 Then F8.Name = XArr(6): F8.Type = DB_SINGLE: Td1.Fields.Append F8
    If VCtr > 7 Then F9.Name = XArr(7): F9.Type = DB_SINGLE: Td1.Fields.Append F9
    If VCtr > 8 Then F10.Name = XArr(8): F10.Type = DB_SINGLE: Td1.Fields.Append F10
    If VCtr > 9 Then F11.Name = XArr(9): F11.Type = DB_SINGLE: Td1.Fields.Append F11
    If VCtr > 10 Then F12.Name = XArr(10): F12.Type = DB_SINGLE: Td1.Fields.Append F12
    If VCtr > 11 Then F13.Name = XArr(11): F13.Type = DB_SINGLE: Td1.Fields.Append F13
    If VCtr > 12 Then F14.Name = XArr(12): F14.Type = DB_SINGLE: Td1.Fields.Append F14
```

```
    If VCtr > 13 Then F15.Name = XArr(13): F15.Type = DB_SINGLE: Td1.Fields.Append F15
    If VCtr > 14 Then F16.Name = XArr(14): F16.Type = DB_SINGLE: Td1.Fields.Append F16
    If VCtr > 15 Then F17.Name = XArr(15): F17.Type = DB_SINGLE: Td1.Fields.Append F17
    If VCtr > 16 Then F18.Name = XArr(16): F18.Type = DB_SINGLE: Td1.Fields.Append F18
    If VCtr > 17 Then F19.Name = XArr(17): F19.Type = DB_SINGLE: Td1.Fields.Append F19
    If VCtr > 18 Then F20.Name = XArr(18): F20.Type = DB_SINGLE: Td1.Fields.Append F20
    If VCtr > 19 Then F21.Name = XArr(19): F21.Type = DB_SINGLE: Td1.Fields.Append F21
    If VCtr > 20 Then F22.Name = XArr(20): F22.Type = DB_SINGLE: Td1.Fields.Append F22

    I1.Name = "I_SECONDS": I1.Fields = "SECONDS": I1.Primary = True: I1.Unique = True: Td1.Index
es.Append I1
    If VArr(0) <> "" Then I2.Name = "I_" & XArr(0): I2.Fields = XArr(0): I2.Primary = False: I2.
Unique = False: Td1.Indexes.Append I2
    If VCtr > 1 Then I3.Name = "I_" & XArr(1): I3.Fields = XArr(1): I3.Primary = False: I3.Uniqu
e = False: Td1.Indexes.Append I3
    If VCtr > 2 Then I4.Name = "I_" & XArr(2): I4.Fields = XArr(2): I4.Primary = False: I4.Uniqu
e = False: Td1.Indexes.Append I4
    If VCtr > 3 Then I5.Name = "I_" & XArr(3): I5.Fields = XArr(3): I5.Primary = False: I5.Uniqu
e = False: Td1.Indexes.Append I5
    If VCtr > 4 Then I6.Name = "I_" & XArr(4): I6.Fields = XArr(4): I6.Primary = False: I6.Uniqu
e = False: Td1.Indexes.Append I6
    If VCtr > 5 Then I7.Name = "I_" & XArr(5): I7.Fields = XArr(5): I7.Primary = False: I7.Uniqu
e = False: Td1.Indexes.Append I7
    If VCtr > 6 Then I8.Name = "I_" & XArr(6): I8.Fields = XArr(6): I8.Primary = False: I8.Uniqu
e = False: Td1.Indexes.Append I8
    If VCtr > 7 Then I9.Name = "I_" & XArr(7): I9.Fields = XArr(7): I9.Primary = False: I9.Uniqu
e = False: Td1.Indexes.Append I9
    If VCtr > 8 Then I10.Name = "I_" & XArr(8): I10.Fields = XArr(8): I10.Primary = False: I10.U
nique = False: Td1.Indexes.Append I10
    If VCtr > 9 Then I11.Name = "I_" & XArr(9): I11.Fields = XArr(9): I11.Primary = False: I11.U
nique = False: Td1.Indexes.Append I11
    If VCtr > 10 Then I12.Name = "I_" & XArr(10): I12.Fields = XArr(10): I12.Primary = False: I1
2.Unique = False: Td1.Indexes.Append I12
    If VCtr > 11 Then I13.Name = "I_" & XArr(11): I13.Fields = XArr(11): I13.Primary = False: I1
3.Unique = False: Td1.Indexes.Append I13
    If VCtr > 12 Then I14.Name = "I_" & XArr(12): I14.Fields = XArr(12): I14.Primary = False: I1
4.Unique = False: Td1.Indexes.Append I14
    If VCtr > 13 Then I15.Name = "I_" & XArr(13): I15.Fields = XArr(13): I15.Primary = False: I1
5.Unique = False: Td1.Indexes.Append I15
    If VCtr > 14 Then I16.Name = "I_" & XArr(14): I16.Fields = XArr(14): I16.Primary = False: I1
6.Unique = False: Td1.Indexes.Append I16
    If VCtr > 15 Then I17.Name = "I_" & XArr(15): I17.Fields = XArr(15): I17.Primary = False: I1
7.Unique = False: Td1.Indexes.Append I17
    If VCtr > 16 Then I18.Name = "I_" & XArr(16): I18.Fields = XArr(16): I18.Primary = False: I1
8.Unique = False: Td1.Indexes.Append I18
    If VCtr > 17 Then I19.Name = "I_" & XArr(17): I19.Fields = XArr(17): I19.Primary = False: I1
9.Unique = False: Td1.Indexes.Append I19
    If VCtr > 18 Then I20.Name = "I_" & XArr(18): I20.Fields = XArr(18): I20.Primary = False: I2
0.Unique = False: Td1.Indexes.Append I20
    If VCtr > 19 Then I21.Name = "I_" & XArr(19): I21.Fields = XArr(19): I21.Primary = False: I2
1.Unique = False: Td1.Indexes.Append I21
    If VCtr > 20 Then I22.Name = "I_" & XArr(20): I22.Fields = XArr(20): I22.Primary = False: I2
2.Unique = False: Td1.Indexes.Append I22

    NewDB.TableDefs.Append Td1

    '------
    'NewDB.Close
    Exit Sub
EH_CreateNewDB:
    MsgBox "Error in CreateNewDB: " & Error$(Err)
    Exit Sub

End Sub

Function CreatePath(ByVal destPath$) As Integer
'----------------------------------------
' Create the path contained in DestPath$
' First char must be drive letter, followed by
' a ":\" followed by the path, if any.
```

```
'-----------------------------------------

    screen.MousePointer = 11

    '-----------------------------------------
    ' Add slash to end of path if not there already
    '-----------------------------------------
    If Right$(destPath$, 1) <> "\" Then
        destPath$ = destPath$ + "\"
    End If


    '-----------------------------------
    ' Change to the root dir of the drive
    '-----------------------------------
    On Error Resume Next
    ChDrive destPath$
    If Err <> 0 Then GoTo errorOut
    ChDir "\"

    '-------------------------------------------------
    ' Attempt to make each directory, then change to it
    '-------------------------------------------------
    BackPos = 3
    forePos = InStr(4, destPath$, "\")
    Do While forePos <> 0
        Tempp$ = Mid$(destPath$, BackPos + 1, forePos - BackPos - 1)

        Err = 0
        MkDir Tempp$
        If Err <> 0 And Err <> 75 Then GoTo errorOut

        Err = 0
        ChDir Tempp$
        If Err <> 0 Then GoTo errorOut

        BackPos = forePos
        forePos = InStr(BackPos + 1, destPath$, "\")
    Loop

    CreatePath = True
    screen.MousePointer = 0
    Exit Function

errorOut:
    MsgBox "Error While Attempting to Create Directories on Destination Drive."
    CreatePath = False
    screen.MousePointer = 0

End Function

Sub DBCompact(DBPath As String)   'fed a path to a database file
    DbStub$ = Left$(DBPath, Len(DBPath) - 3)
    DbNew$ = DbStub$ & "new"
    DBOld$ = DbStub$ & "mdb"
    'if there is enough free diskspace, compact database
    If FreeDiskSpace(Left$(DBPath, 1)) > Len(DBPath) Then
        CompactDatabase DBOld$, DbNew$
        Kill DBOld$
        Name DbNew$ As DBOld$
    Else
        MsgBox "Not enough free disk space to compact database"
    End If
End Sub

Sub DBOpen(DBF$)
    Dim RtnString As String * 64
    f_about.Show
    f_about.Timer1.Enabled = True
```

```
    ClearCurrentDB
    x% = InStr(DBF$, "\")
    Do While x%
        ct% = x%
        x% = InStr(ct% + 1, DBF$, "\")
    Loop
    DBPath$ = Left$(DBF$, ct%)
    DBFile$ = Right$(DBF$, Len(DBF$) - ct%)
    f_data.Caption = "Responses: " & Left$(DBFile$, Len(DBFile$) - 4)

    'locate AVI's from INI file
    DBIniFile$ = DBPath$ & Left$(DBFile$, Len(DBFile$) - 3) & "ini"
    di% = GetPrivateProfileString("Main", "AVIPath", "", RtnString, 63, DBIniFile$)
    AVIPath$ = FixAPIString(RtnString)
    LoadVideoFiles2 AVIPath$
    If NumAVIs = 0 Then
        Beep
        MsgBox "Database cannot be opened: no video files."
        f_about.Timer1.Enabled = False
        Unload f_about
        Exit Sub
    End If

    'fill variable list from database
    Set NewDB = OpenDatabase(DBF$, True, False)
    Set TblVar = NewDB.OpenTable("T_DATA")
    For i = 1 To TblVar.Fields.Count - 1
        f_query.List1.AddItem UCase$(TblVar.Fields(i).Name)
    Next i

    AddGraphSeconds

    'Get AVILeadin, AVILeadout
    di% = GetPrivateProfileString("Main", "AVILeadin", "", RtnString, 63, DBIniFile$)
    AVILeadin = Val(FixAPIString(RtnString))
    di% = GetPrivateProfileString("Main", "AVILeadout", "", RtnString, 63, DBIniFile$)
    AVILeadout = Val(FixAPIString(RtnString))

    f_about.Timer1.Enabled = False
    Unload f_about

End Sub

Function DeviceColors() As Long
    FhDC% = GetDC(f_main.hWnd)
    DeviceColors = GetDeviceCaps(FhDC%, 14) * 2 ^ GetDeviceCaps(FhDC%, 12)
End Function

Sub DrawGraph()
    TblVar.Index = "I_SECONDS"
    TblVar.MoveLast
    f_data.pnlGraph.ScaleWidth = TblVar("SECONDS")
    BarInc = Int(f_data.pnlGraph.ScaleWidth / 10)
    'TblVar.MoveFirst

    If MaxY1 = 10 Then GTop = 10 Else GTop = Int(MaxY1) + 1
    GBot = Int(MinY1)
    f_data.pnlGraph.ScaleHeight = GTop - GBot
    f_data.Label2.Caption = Format$(GTop)
    f_data.Label3.Caption = Format$(GBot)
    x% = 0
    Ds1.MoveFirst
    'f_main.pnlStatus.FloodType = 1
    'f_main.pnlStatus.FloodPercent = 0
    f_data.pnlGraph.CurrentX = 0
    f_data.pnlGraph.CurrentY = GTop - Ds1(NewFld$)
    Do While Not (Ds1.EOF)
        f_data.pnlGraph.Line -(x%, GTop - Ds1(NewFld$)), RED
        Ds1.MoveNext
        'If x% Mod BarInc = 0 Then f_main.pnlStatus.FloodPercent = Int(x% / f_data.pnlGraph.Scal
```

```
eWidth * 100)
        If x% Mod BarInc = 0 Then DoEvents
        x% = x% + 1
    Loop
    f_data.pnlGraph.CurrentX = 0
    f_data.pnlGraph.CurrentY = 0
    '---!!!--- f_data.pnlGraph.Print "StDev=" & Format$(MyStdev, "0.00") & "  Var=" & Format$(My
Var, "0.00")
    'f_main.pnlStatus.FloodType = 0

    'Place max and min pointers
    f_data.MaxPointer.Left = 375 + Int(MaxX1 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Widt
h)
    f_data.MaxPointer2.Left = 375 + Int(MaxX2 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Wid
th)
    f_data.MaxPointer3.Left = 375 + Int(MaxX3 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Wid
th)
    f_data.MinPointer.Left = 375 + Int(MinX1 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Widt
h)
    f_data.MinPointer2.Left = 375 + Int(MinX2 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Wid
th)
    f_data.MinPointer3.Left = 375 + Int(MinX3 / f_data.pnlGraph.ScaleWidth * f_data.pnlGraph.Wid
th)
    f_data.LeadinPointer.Left = 375 + Int(AVILeadin / f_data.pnlGraph.ScaleWidth * f_data.pnlGra
ph.Width)
    f_data.LeadoutPointer.Left = 375 + f_data.pnlGraph.Width - Int(AVILeadout / f_data.pnlGraph.
ScaleWidth * f_data.pnlGraph.Width)
    f_data.MaxPointer.Visible = True
    f_data.MaxPointer2.Visible = True
    f_data.MaxPointer3.Visible = True
    f_data.MinPointer.Visible = True
    f_data.MinPointer2.Visible = True
    f_data.MinPointer3.Visible = True
    f_data.LeadinPointer.Visible = True
    f_data.LeadoutPointer.Visible = True
End Sub

Function Exists(F$) As Integer
    Dim FStruct As OFSTRUCT
    di% = OPenFile(F$, FStruct, OF_EXIST)
    Exists = di%
End Function

Sub FillGradForm(Target As Form, MyColor As String)
On Error GoTo EH
Dim FormY As Integer, CValue As Integer, DeltaY As Integer
Dim i As Integer, Result As Integer, OldMode As Integer
Dim hBrush As Integer
Dim Region As RECT

STEPS = 255

' Save old scale mode and switch to Pixel mode.

OldMode = Target.ScaleMode
Target.ScaleMode = 3  'Pixel

' Divide the form into STEPS regions.

FormY = Target.ScaleHeight
DeltaY = FormY \ STEPS

' Set coordinates of first region to fill.

Region.Left = 0
Region.Right = Target.ScaleWidth
Region.Top = 0
Region.Bottom = DeltaY

' Starting color.
```

```
CValue = 255

For i = 1 To STEPS
    Select Case MyColor
        Case "red"
            hBrush = CreateSolidBrush(RGB(CValue, 0, 0))
        Case "green"
            hBrush = CreateSolidBrush(RGB(0, CValue, 0))
        Case "blue"
            hBrush = CreateSolidBrush(RGB(0, 0, CValue))
        Case "gray"
            hBrush = CreateSolidBrush(RGB(CValue, CValue, CValue))
    End Select
    Call FillRect(Target.hDC, Region, hBrush)
    di% = DeleteObject(hBrush)
    Region.Top = Region.Bottom
    Region.Bottom = Region.Bottom + DeltaY
    CValue = CValue - 1
Next i

' Fill the remainder of the form with black.

Region.Bottom = Region.Bottom + STEPS
hBrush = CreateSolidBrush(RGB(0, 0, 0))
Call FillRect(Target.hDC, Region, hBrush)
di% = DeleteObject(hBrush)

' Reset the original scale mode.

Target.ScaleMode = OldMode
Exit Sub
EH:
    MsgBox "Error in FillGradForm"
    Exit Sub
End Sub

Sub FindAVIFiles(FileSpec As String, SearchPath As String)
ReDim DirName(0 To 15) As String
Dim DirCount As Integer
Dim Filename As String, Attributes As Integer
Dim x As Integer

If Right$(SearchPath, 1) <> "\" Then SearchPath = SearchPath & "\"
DirCount = 0
Filename = Dir$(SearchPath & FileSpec, ATTR_NORMAL + ATTR_SYSTEM + ATTR_HIDDEN)
Do Until Filename = ""
    TArr(NumAVIs) = SearchPath & Filename
    NumAVIs = NumAVIs + 1
    f_main.pnlStatus.Caption = "Looking for video files on disk " & Left$(SearchPath, 1) & " ...
  found:" & Format$(NumAVIs)
    Filename = Dir$
    DoEvents
    If CancelFlag Then Exit Sub
Loop

Filename = Dir$(SearchPath & "*.*", ATTR_NORMAL + ATTR_SYSTEM + ATTR_HIDDEN + ATTR_DIRECTORY)
Do Until Filename = ""
    If Filename <> "." And Filename <> ".." Then
        Attributes = GetAttr(SearchPath & Filename)
        If (Attributes And ATTR_DIRECTORY) Then
            If DirCount > UBound(DirName) Then
                ReDim Preserve DirName(0 To DirCount + 15)
            End If
            DirName(DirCount) = SearchPath & Filename
            DirCount = DirCount + 1
        End If
    End If
    Filename = Dir$
    DoEvents
```

```
    If CancelFlag Then Exit Sub
Loop
For x = 0 To DirCount - 1
    FindAVIFiles FileSpec, DirName(x)
Next x

End Sub


Sub FindHisLos(NewFld$)
    Dim Ds2 As snapshot
    Dim IsLeadin As Integer
    ExRange% = 10
    IsLeadin = False
    SQLLEAD$ = ""
    TblVar.Index = "I_" & NewFld$

    SQLMAIN$ = "SELECT MAX([" & NewFld$ & "]) AS QMax1, "
    SQLMAIN$ = SQLMAIN$ & "MIN([" & NewFld$ & "]) AS QMin1 FROM T_DATA"

    SQLTOP$ = "SELECT MAX([" & NewFld$ & "]) AS QMax1, MIN([" & NewFld$ & "]) AS QMin1, "
    SQLTOP$ = SQLTOP$ & "STDEV([" & NewFld$ & "]) AS QStdev1, "
    SQLTOP$ = SQLTOP$ & "VAR([" & NewFld$ & "]) AS QVar1 FROM T_DATA"
    If AVILeadin > 0 Then
        SQLTOP$ = SQLTOP$ & " WHERE SECONDS > " & Format$(AVILeadin)
        SQLLEAD$ = " AND SECONDS > " & Format$(AVILeadin)
        IsLeadin = True
    End If
    If AVILeadout > 0 Then
        If IsLeadin Then
            SQLTOP$ = SQLTOP$ & " AND SECONDS < " & Format$(Int(AVILength / 1000) - AVILeadout)
        Else
            SQLTOP$ = SQLTOP$ & " WHERE SECONDS < " & Format$(Int(AVILength / 1000) - AVILeadout
)
        End If
        SQLLEAD$ = SQLLEAD$ & " AND SECONDS < " & Format$(Int(AVILength / 1000) - AVILeadout)
    End If
    Set Ds2 = NewDB.CreateSnapshot(SQLTOP$)
    MaxY1 = Ds2("QMax1")
    'MyCrit$ = "[" & NewFld$ & "] = " & Format$(MaxY1)
    'MyCrit$ = MyCrit$ & SQLLEADIN$
    'Ds1.FindFirst MyCrit$
    'MaxX1 = Ds1("SECONDS")
    TblVar.Seek "=", MaxY1
    MaxX1 = TblVar("SECONDS")
    MinY1 = Ds2("QMin1")
    MyCrit$ = "[" & NewFld$ & "] = " & Format$(MinY1)
    MyCrit$ = MyCrit$ & SQLLEAD$
    Ds1.FindFirst MyCrit$
    MinX1 = Ds1("SECONDS")
    'TblVar.Seek "=", MinY1
    'MinX1 = TblVar("SECONDS")
    MyStdev = Ds2("QStdev1")
    MyVar = Ds2("QVar1")

    SQLMAIN2$ = SQLMAIN$ & " WHERE SECONDS NOT BETWEEN " & Format$(MaxX1 - ExRange%) & " AND " &
 Format$(MaxX1 + ExRange%)
    SQLMAIN2$ = SQLMAIN2$ & " AND SECONDS NOT BETWEEN " & Format$(MinX1 - ExRange%) & " AND " &
Format$(MinX1 + ExRange%)
    SQLMAIN2$ = SQLMAIN2$ & SQLLEAD$
    Set Ds2 = NewDB.CreateSnapshot(SQLMAIN2$)
    MaxY2 = Ds2("QMax1")
    MyCrit$ = "[" & NewFld$ & "] = " & Format$(MaxY2)
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MaxX1 - ExRange%) & " AND " & F
at$(MaxX1 + ExRange%)
    MyCrit$ = MyCrit$ & SQLLEAD$
    Ds1.FindFirst MyCrit$
    MaxX2 = Ds1("SECONDS")
    MinY2 = Ds2("QMin1")
    MyCrit$ = "[" & NewFld$ & "] = " & Format$(MinY2)
```

```
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MinX1 - ExRange%) & " AND " & Form
at$(MinX1 + ExRange%)
    MyCrit$ = MyCrit$ & SQLLEAD$
    Ds1.FindFirst MyCrit$
    MinX2 = Ds1("SECONDS")

    SQLMAIN3$ = SQLMAIN$ & " WHERE SECONDS NOT BETWEEN " & Format$(MaxX1 - ExRange%) & " AND " &
 Format$(MaxX1 + ExRange%)
    SQLMAIN3$ = SQLMAIN3$ & " AND SECONDS NOT BETWEEN " & Format$(MaxX2 - ExRange%) & " AND " &
Format$(MaxX2 + ExRange%)
    SQLMAIN3$ = SQLMAIN3$ & " AND SECONDS NOT BETWEEN " & Format$(MinX1 - ExRange%) & " AND " &
Format$(MinX1 + ExRange%)
    SQLMAIN3$ = SQLMAIN3$ & " AND SECONDS NOT BETWEEN " & Format$(MinX2 - ExRange%) & " AND " &
Format$(MinX2 + ExRange%)
    SQLMAIN3$ = SQLMAIN3$ & SQLLEAD$
    Set Ds2 = NewDB.CreateSnapshot(SQLMAIN3$)
    MaxY3 = Ds2("QMax1")
    MyCrit$ = "[" & NewFld$ & "] = " & Format$(MaxY3)
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MaxX1 - ExRange%) & " AND " & Form
at$(MaxX1 + ExRange%)
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MaxX2 - ExRange%) & " AND " & Form
at$(MaxX2 + ExRange%)
    MyCrit$ = MyCrit$ & SQLLEAD$
    Ds1.FindFirst MyCrit$
    MaxX3 = Ds1("SECONDS")
    MinY3 = Ds2("QMin1")
    MyCrit$ = "[" & NewFld$ & "] = " & Format$(MinY3)
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MinX1 - ExRange%) & " AND " & Form
at$(MinX1 + ExRange%)
    MyCrit$ = MyCrit$ & " AND SECONDS NOT BETWEEN " & Format$(MinX2 - ExRange%) & " AND " & Form
at$(MinX2 + ExRange%)
    MyCrit$ = MyCrit$ & SQLLEAD$
    Ds1.FindFirst MyCrit$
    MinX3 = Ds1("SECONDS")
    Ds2.Close

    'write the results to the INI
    NS$ = Format$(MaxX1) & "/" & Format$(MaxX2) & "/" & Format$(MaxX3) & "/"
    NS$ = NS$ & Format$(MinX1) & "/" & Format$(MinX2) & "/" & Format$(MinX3) & "/"
    NS$ = NS$ & Format$(MaxY1) & "/" & Format$(MinY1)
    x% = WritePrivateProfileString("Main", NewFld$, NS$, DBIniFile$)
    If x% = 0 Then
        fh% = FreeFile
        Open DBIniFile$ For Append As fh%
        Print #fh%, NewFld$ & "=" & NS$
        Close #fh%
    End If
End Sub

Function FixAPIString$(ByVal test$)
    FixAPIString$ = Trim(Left$(test$, InStr(test$, Chr$(0)) - 1))
End Function

Function FixFieldName(S$) As String
    i = InStr(S$, ".")
    Do While i > 0
        S$ = Left$(S$, i - 1) & Right$(S$, Len(S$) - i)
        i = InStr(S$, ".")
    Loop
    i = InStr(S$, "=")
    Do While i > 0
        S$ = Left$(S$, i - 1) & Right$(S$, Len(S$) - i)
        i = InStr(S$, "=")
    Loop
    FixFieldName = S$
End Function

Sub FormExplode(Target As Form)

' "explodes" a form by drawing successively larger rectangles.
```

```
' using the form's background color, to fill the form area.
' Should be called from the Form_Load event procedure.

' Number of steps to use in expanding the rectangle. More steps
' result in a slower but smoother "explosion."

Const STEPS = 40

Dim FormWidth As Integer, FormHeight As Integer
Dim Count As Integer, x As Integer, Y As Integer
Dim XStep As Integer, YStep As Integer
Dim hDCScreen As Integer, hBrush As Integer
Dim MyRect As RECT

' Get the form's coordinates and detemine its height and width.

Call GetWindowRect(Target.hWnd, MyRect)

FormWidth = MyRect.Right - MyRect.Left
FormHeight = MyRect.Bottom - MyRect.Top

' Get the screen's device context.

hDCScreen = GetDC(0)

' Create a solid brush that uses the form's background color.

hBrush = CreateSolidBrush(Target.BackColor)
di% = SelectObject(hDCScreen, hBrush)

' Draw successively larger rectangles until the form's
' entire area is filled.

For Count = 1 To STEPS
    XStep = FormWidth * (Count / STEPS)
    YStep = FormHeight * (Count / STEPS)
    x = MyRect.Left + (FormWidth - XStep) / 2
    Y = MyRect.Top + (FormHeight - YStep) / 2
    Call Rectangle(hDCScreen, x, Y, x + XStep, Y + YStep)
Next Count

' Release the device context and brush, and display the form.

di% = ReleaseDC(0, hDCScreen)
di% = DeleteObject(hBrush)
Target.Visible = True

End Sub

Function FormLoaded(MyForm As String)
    FormLoaded = False
    For x = 0 To Forms.Count - 1
        If Forms(x).Tag = MyForm Then
            FormLoaded = True
            Exit Function
        End If
    Next x
End Function

Function FreeDiskSpace(Drive As String) As Long
    OldDir$ = Left$(CurDir$, 1)
    ChDrive Drive
    FreeDiskSpace = DiskSpaceFree()
    ChDrive OldDir$
End Function

Function GetAviSize(AVIPath As String) As Long

    On Error GoTo EH_GetAviSize
```

```
    'GetAviSize=
    Exit Function
EH_GetAviSize:
    MsgBox "Error in GetAviSize: " & Error(Err)
    Exit Function
End Function

Function GetHisLosFromINI%(VN$)
    Dim RtnString As String * 128
    di% = GetPrivateProfileString("Main", VN$, "", RtnString, 127, DBIniFile$)
    If di% = 0 Then GetHisLosFromINI = 0: Exit Function Else GetHisLosFromINI = 1
    DS$ = FixAPIString(RtnString)
    x = InStr(DS$, "/")
    MaxX1 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MaxX2 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MaxX3 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MinX1 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MinX2 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MinX3 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    x = InStr(DS$, "/")
    MaxY1 = Left$(DS$, x - 1)
    DS$ = Mid$(DS$, x + 1)
    MinY1 = DS$
End Function

Sub GetSysInfo()
Dim x As Long, DOSVer As Long
Dim WinMajor As Integer, WinMinor As Integer
Dim DOSMajor As Integer, DOSMinor As Long
Dim i As Integer

' Get DOS and Windows versions.
x = GetVersion()
DOSVer = x \ &H10000
WinMajor = x And &HFF
WinMinor = (x And &HFFF) \ 256
DOSMinor = DOSVer And &HFF
DOSMajor = DOSVer \ 256
DOSVersion$ = "DOS version " & Str$(DOSMajor) & "." & Right$(Str$(DOSMinor), Len(Str$(DOSMinor))
 - 1)
WinVersion$ = "Windows version " & Str$(WinMajor) & "." & Right$(Str$(WinMinor), Len(Str$(WinMin
or)) - 1)

' Get free memory.
Free$ = "Free memory =" & Str$(GetFreeSpace(i)) & " bytes."

' Get the Windows status flags.
x = GetWinFlags()

' Get CPU type.
If x And WF_CPU286 Then
    CPU$ = "CPU: 80286"
ElseIf x And WF_CPU386 Then
    CPU$ = "CPU: 80386"
ElseIf x And WF_CPU486 Then
    CPU$ = "CPU: 80486"
Else
    CPU$ = "CPU: Unknown"
```

```
End If

' co-processor present?
If x And WF_80x87 Then
    MathCO$ = "Math co-processor: present"
Else
    MathCO$ = "Math co-processor: not present"
End If

' EMS Frame type.
If x And WF_LARGEFRAME Then
    EMS$ = "EMS frame: large"
Else
    EMS$ = "EMS frame: small"
End If

' Mode.
If x And WF_ENHANCED Then
    Mode$ = "Windows mode: enhanced"
Else
    Mode$ = "Windows mode: standard"
End If

' Get keyboard type.
i = GetKeyboardType(0)
Select Case i
    Case 1
        KT$ = "IBM PC/XT, or compatible (83-key) keyboard"
    Case 2
        KT$ = "Olivetti ICO (102-key) keyboard"
    Case 3
        KT$ = "IBM AT (84-key) or similar keyboard"
    Case 4
        KT$ = "IBM Enhanced (101- or 102-key) keyboard"
    Case 5
        KT$ = "Nokia 1050 or similar keyboard"
    Case 6
        KT$ = "Nokia 9140 or similar keyboard"
    Case 7
        KT$ = "Japanese keyboard"
    Case Else
        KT$ = "Unknown keyboard type"
End Select

' Number of function keys.
i = GetKeyboardType(2)

End Sub

Function GetSystemDir$()
    Dim Sys As String * 256
    x = GetSystemDirectory(Sys, Len(Sys))
    x = InStr(1, Sys, Chr$(0))
    GetSystemDir$ = Left$(Sys, InStr(Sys, Chr$(0)) - 1)
End Function

Function GetVolume() As Integer
    mcihwnd = dwGetControlHwnd(f_video.MCIWnd1(0))
    di% = SendMessage(mcihwnd, MCIWNDM_GETVOLUME, 0, 0)
    GetVolume = Int(di%)
End Function

Sub GraphClick(x As Single)
    TmpMode$ = ""
    If VMode = "Play" Then
        TmpMode$ = "Play"
        VideoStop
    End If
    TmpPos = Int(AVILength * x / f_data.pnlGraph.ScaleWidth)
    TmpTot = 0
```

```
    i = 0
    TmpTot = f_video.MCIWnd1(0).Length
    Do While TmpPos >= TmpTot
        i = i + 1
        TmpTot = TmpTot + f_video.MCIWnd1(i).Length
    Loop
    TmpCurAVI = i

    If TmpCurAVI <> CurAVI Then
        f_video.MCIWnd1(CurAVI).Visible = False
        CurAVI = TmpCurAVI
        SetPrevAVITot
    End If
    f_video.MCIWnd1(CurAVI).Position = TmpPos - PrevAVITot
    DoEvents
    f_video.MCIWnd1(CurAVI).Visible = True
    If TmpMode$ = "Play" Then
        VideoPlay
    End If
End Sub

Sub ImportDIF(DBF$, MyDB$)
    Dim FileNum As Integer
    Dim DArr() As String * 3

    'find root of DBF$
    x = Len(DBF$)
    Do While InStr("\:", Mid$(DBF$, x, 1)) = 0
        x = x - 1
    Loop
    DBPath$ = Left$(DBF$, x)
    DBFilePat$ = Mid$(DBF$, x + 1, Len(DBF$) - x - 5)

    'On Error GoTo EH_ImportData
    'how many DIF files are there?
    f_main.pnlStatus.Caption = "Counting DIF files ..."
    NumDifs% = 0
    DIFFilter$ = Left$(DBF$, Len(DBF$) - 5) & "*.dif"
    FN$ = Dir$(DIFFilter$, ATTR_NORMAL Or ATTR_HIDDEN Or ATTR_SYSTEM Or ATTR_DIRECTORY)
    Do While FN$ <> ""
        NumDifs% = NumDifs% + 1
        FN$ = Dir$
    Loop

    'copy file to string
    Open DBF$ For Input As 1
    S$ = Input$(LOF(1), 1)
    Close 1

    'identify variable names and put into array
    f_main.pnlStatus.Caption = "Identifying variables ..."
    VCtr = 0
    S$ = Mid$(S$, InStr(S$, "Sec 59") + 8)
    q$ = S$
    Do
        S$ = Mid$(S$, InStr(S$, Chr(34)) + 1)
        X1 = InStr(S$, Chr(34))
        T$ = Left$(S$, X1 - 1)
        VCtr = VCtr + 1
        S$ = Mid$(S$, X1 + 1)
    Loop Until InStr(S$, Chr(34)) = 0
    ReDim VArr(VCtr)
    ReDim XArr(VCtr)
    S$ = q$
    NN = 0
    Do
        S$ = Mid$(S$, InStr(S$, Chr(34)) + 1)
        X1 = InStr(S$, Chr(34))
        T$ = Left$(S$, X1 - 1)
        VArr(NN) = T$
```

```
        XArr(NN) = FixFieldName(T$)
        'f_query.List1.AddItem UCase$(FixFieldName(T$))
        NN = NN + 1
        S$ = Mid$(S$, X1 + 1)
    Loop Until InStr(S$, Chr(34)) = 0


    'build database file
    f_main.pnlStatus.Caption = "Building database ..."
    CreateNewModDB MyDB$
    ReDim DArr(NumDifs% * 60, VCtr)

    'read data from files into array
    For n = 0 To NumDifs% - 1
        FN$ = DBFilePat$ & Format$(n) & ".DIF"
        f_main.pnlStatus.Caption = "Processing file " & FN$ & " ..."
        Open DBPath$ & FN$ For Input As 1
        S$ = Input$(LOF(1), 1)
        Close 1
        For m = 0 To VCtr - 1
            S$ = Mid$(S$, InStr(S$, Chr(34) & VArr(m) & Chr(34)) + Len(VArr(m)) + 5)
            For r = 0 To 59
                DArr((n) * 60 + r, m) = Mid$(S$, 1 + r * 8, 3)
            Next r
        Next m
    Next n

    'insert data from array into database
    Set TblVar = NewDB.OpenTable("T_DATA")
    For n = 0 To NumDifs% * 60 - 1
        If n Mod 10 = 0 Then f_main.pnlStatus.Caption = "Processing second " & Format$(n)
        TblVar.AddNew
        TblVar("SECONDS") = n
        For m = 0 To VCtr - 1
            TblVar(XArr(m)) = Val(DArr(n, m))
        Next m
        TblVar.Update
    Next n

    Close 1
    'find highs and lows for each variable
    For m = 0 To VCtr - 1
        NewFld$ = UCase$(XArr(m))
        SQLST$ = "SELECT SECONDS, [" & NewFld$ & "] FROM T_DATA"
        Set Ds1 = NewDB.CreateSnapshot(SQLST$)
        f_main.pnlStatus.Caption = "Finding highs and lows for " & NewFld$
        FindHisLos NewFld$
        Ds1.Close
    Next m

    Exit Sub
EH_ImportDataCancel:
    Exit Sub

EH_ImportData:
    If Err <> 5 Then
        MsgBox "Error in ImportData:" & Error(Err)
    End If
    Exit Sub

End Sub

Function IsBlankFloppy(destPath$) As Integer
    IsBlankFloppy = True
    FN$ = Dir$(destPath$ & "*.*", ATTR_NORMAL Or ATTR_HIDDEN Or ATTR_SYSTEM Or ATTR_DIRECTORY)
    Do While FN$ <> ""
        MsgBox FN$
        If FN$ <> "." And FN$ <> ".." Then
            IsBlankFloppy = False
            Exit Do
```

```
        End If
        FN$ = Dir$
    Loop
End Function

Function IsRemoveableDrive(Drive As String) As Integer
    di% = GetDriveType(InStr("abcdefghijklmnopqrstuvwxyz", LCase$(Left$(Drive, 1))) - 1)
    If di% = 2 Then IsRemoveableDrive = True Else IsRemoveableDrive = False
End Function

Function IsValidPath(destPath$, ByVal DefaultDrive$) As Integer

    '----------------------------
    ' Remove left and right spaces
    '----------------------------
    destPath$ = RTrim$(LTrim$(destPath$))


    '----------------------------
    ' Check Default Drive Parameter
    '----------------------------
    If Right$(DefaultDrive$, 1) <> ":" Or Len(DefaultDrive$) <> 2 Then
        MsgBox "Bad default drive parameter specified in IsValidPath Function.  You passed, """
 + DefaultDrive$ + """.  Must be one drive letter and "":"".  For example, ""C:"", ""D:""..."
        GoTo parseErr
    End If


    '-----------------------------------------------------
    ' Insert default drive if path begins with root backslash
    '-----------------------------------------------------
    If Left$(destPath$, 1) = "\" Then
        destPath$ = DefaultDrive + destPath$
    End If

    '----------------------------
    ' check for invalid characters
    '----------------------------
    On Error Resume Next
    tmp$ = Dir$(destPath$)
    If Err <> 0 Then
        GoTo parseErr
    End If


    '---------------------------------------
    ' Check for wildcard characters and spaces
    '---------------------------------------
    If (InStr(destPath$, "*") <> 0) Then GoTo parseErr
    If (InStr(destPath$, "?") <> 0) Then GoTo parseErr
    If (InStr(destPath$, " ") <> 0) Then GoTo parseErr


    '---------------------------------------
    ' Make Sure colon is in second char position
    '---------------------------------------
    If Mid$(destPath$, 2, 1) <> Chr$(58) Then GoTo parseErr


    '------------------------------
    ' Insert root backslash if needed
    '------------------------------
    If Len(destPath$) > 2 Then
      If Right$(Left$(destPath$, 3), 1) <> "\" Then
        destPath$ = Left$(destPath$, 2) + "\" + Right$(destPath$, Len(destPath$) - 2)
      End If
    End If

    '------------------------
    ' Check drive to install on
```

```
    '-------------------------
    Drive$ = Left$(destPath$, 1)
    ChDrive (Drive$)                                                    ' Try to change to t
he dest drive
    If Err <> 0 Then GoTo parseErr

    '-----------
    ' Add final \
    '-----------
    If Right$(destPath$, 1) <> "\" Then
        destPath$ = destPath$ + "\"
    End If


    '-------------------------------------
    ' Root dir is a valid dir
    '-------------------------------------
    If Len(destPath$) = 3 Then
        If Right$(destPath$, 2) = ":\" Then
            GoTo ParseOK
        End If
    End If


    '------------------------
    ' Check for repeated Slash
    '------------------------
    If InStr(destPath$, "\\") <> 0 Then GoTo parseErr

    '-------------------------------------
    ' Check for illegal directory names
    '-------------------------------------
    legalChar$ = "!#$%&'()-0123456789@ABCDEFGHIJKLMNOPQRSTUVWXYZ^_`{}~."
    BackPos = 3
    forePos = InStr(4, destPath$, "\")
    Do
        Tempp$ = Mid$(destPath$, BackPos + 1, forePos - BackPos - 1)

        '----------------------------
        ' Test for illegal characters
        '----------------------------
        For i = 1 To Len(Tempp$)
            If InStr(legalChar$, UCase$(Mid$(Tempp$, i, 1))) = 0 Then GoTo parseErr
        Next i

        '-----------------------------------------
        ' Check combinations of periods and lengths
        '-----------------------------------------
        periodPos = InStr(Tempp$, ".")
        Length = Len(Tempp$)
        If periodPos = 0 Then
            If Length > 8 Then GoTo parseErr                            ' Base too long
        Else
            If periodPos > 9 Then GoTo parseErr                         ' Base too long
            If Length > periodPos + 3 Then GoTo parseErr                ' Extension too long
            If InStr(periodPos + 1, Tempp$, ".") <> 0 Then GoTo parseErr ' Two periods not allow
ed
        End If

        BackPos = forePos
        forePos = InStr(BackPos + 1, destPath$, "\")
    Loop Until forePos = 0

ParseOK:
    IsValidPath = True
    Exit Function

parseErr:
    IsValidPath = False
End Function
```

```
Sub LaunchLastDB()
    Dim RtnString As String * 128
    screen.MousePointer = 11
    ModIni$ = app.Path & "\modular.ini"
    di% = GetPrivateProfileString("Main", "LastDB", "", RtnString, 127, ModIni$)
    CurDBPath = FixAPIString(RtnString)
    If Len(CurDBPath) Then DBOpen CurDBPath
    screen.MousePointer = 0
End Sub

Sub LoadVideoFiles2(DBF$)    'ex: "clinton*.avi"
    Dim x, Y, z, n As Integer
    Dim RtnString As String * 16
    'On Error GoTo EH_LoadVideoFiles2

    'unload all but first mciwndx
    i = 0
    Do
        If f_video.Controls(i).Tag = "newmciwnd" Then Unload f_video.Controls(i) Else i = i + 1
    Loop Until i >= f_video.Controls.Count
    f_video.MCIWnd1(0).Filename = ""
    f_video.pnlVidFrame.Visible = False
    DoEvents
    NumAVIs = 0

    'find all .avi files and put them into TArr
    DBIniFile$ = DBPath$ & Left$(DBFile$, Len(DBFile$) - 3) & "ini"
    di% = GetPrivateProfileString("Main", "AVIDrives", "", RtnString, 16, DBIniFile$)
    AVIDrives$ = FixAPIString(RtnString)
    If AVIDrives$ = "" Then
        Beep
        Msg$ = "Enter drive letters to be searched for video files (ex: CDGK):"
        AVIDrives$ = InputBox$(Msg$, "Select Drive(s) For Video Files")
        di% = WritePrivateProfileString("Main", "AVIDrives", AVIDrives$, DBIniFile$)
    End If
    For x = 1 To Len(AVIDrives$)
        D$ = Mid$(AVIDrives$, x, 1)
        f_main.pnlStatus.Caption = "Looking for video files on disk " & D$ & " ..."
        FindAVIFiles DBF$, D$ & ":\"
    Next x

    'what if no AVI's are found?
    Do While NumAVIs = 0
        Msg$ = "No video files were found on the following drive(s): " & AVIDrives$
        Msg$ = Msg$ & "   Do you want to look on other drive(s)?"
        If MsgBox(Msg$, 52, "No Video Files Found") = 6 Then
            Msg$ = "Enter drive letters to be searched for video files (ex: CDGK):"
            AVIDrives$ = InputBox$(Msg$, "Select Drive(s) For Video Files")
            di% = WritePrivateProfileString("Main", "AVIDrives", AVIDrives$, DBIniFile$)
            For x = 1 To Len(AVIDrives$)
                D$ = Mid$(AVIDrives$, x, 1)
                f_main.pnlStatus.Caption = "Looking for video files on disk " & D$ & " ..."
                FindAVIFiles DBF$, D$ & ":\"
            Next x
        Else
            Exit Sub
        End If
    Loop

    'move elements from TArr to VidArr
    ReDim VidArr(NumAVIs)
    For x = 0 To NumAVIs - 1
        T$ = TArr(x)
        Y = Val(Mid$(T$, Len(T$) - 4, 1)) - 1
        VidArr(Y) = T$
    Next x

    'load the .avi's from VidArr into the controls
    f_main.pnlStatus.Caption = "Initializing video file 1: " & VidArr(0)
```

```
    f_video.MCIWnd1(0).Filename = VidArr(0)
    DoEvents
    WR1 = f_video.MCIWnd1(0).Width / (f_video.Width - 345)
    HR1 = f_video.MCIWnd1(0).Height / (f_video.Height - 1220)
    If WR1 > HR1 Then
        f_video.MCIWnd1(0).Zoom = 100 / WR1
    Else
        f_video.MCIWnd1(0).Zoom = 100 / HR1
    End If
    DoEvents

    f_video.pnlVidFrame.Move 0, 0, f_video.MCIWnd1(0).Width + 240, f_video.MCIWnd1(0).Height + 2
40
    f_video.pnlVidFrame.Visible = True
    f_video.MCIWnd1(0).TimeFormat = "milliseconds"
    AVILength = f_video.MCIWnd1(0).Length

    DoEvents
    For x = 1 To NumAVIs - 1
        f_main.pnlStatus.Caption = "Initializing video file " & Format$(x + 1) & ": " & VidArr(x
)
        Load f_video.MCIWnd1(x)
        f_video.MCIWnd1(x).Visible = False
        f_video.MCIWnd1(x).Playbar = False
        f_video.MCIWnd1(x).AutosizeWindow = True
        f_video.MCIWnd1(x).Filename = VidArr(x)
        f_video.MCIWnd1(x).Move f_video.MCIWnd1(0).Left, f_video.MCIWnd1(0).Top

        f_video.MCIWnd1(x).Zoom = f_video.MCIWnd1(0).Zoom
        f_video.MCIWnd1(x).Tag = "newmciwnd"
        f_video.MCIWnd1(x).TimeFormat = "milliseconds"
        f_video.MCIWnd1(x).TimerFreq = 500
        f_video.MCIWnd1(x).WantPosEvent = True
        AVILength = AVILength + f_video.MCIWnd1(x).Length
    Next x
    CurAVI = 0
    PrevAVITot = 0
    VMode = "Stop"
    f_main.pnlStatus.Caption = "Video setup complete"
    Exit Sub
EH_LoadVideoFiles2:
    screen.MousePointer = 0
    If Err <> 5 Then
        MsgBox "Error in LoadVideoFiles2: " & Error(Err)
    End If
    Exit Sub
End Sub

Sub PlayNote(Sound As String)
    Const S_NORMAL = 0
    Const S_LEGATO = 1
    Const S_STACCATO = 2
    di% = SetVoiceQueueSize(1, 512)
    If G_SoundOn = True Then
        Select Case Sound
            Case "Confirm"
                'di% = SetVoiceAccent(1, 120, 60, S_LEGATO, 0)
                di% = SetVoiceNote(1, 24, 32, 1)
                di% = SetVoiceNote(1, 30, 32, 1)
            Case "Warn"
                'di% = SetVoiceAccent(1, 120, 60, S_LEGATO, 0)
                di% = SetVoiceNote(1, 30, 32, 1)
                di% = SetVoiceNote(1, 24, 32, 1)
            Case "Error"
                'di% = SetVoiceAccent(1, 120, 60, S_LEGATO, 0)
                di% = SetVoiceNote(1, 48, 32, 1)
                di% = SetVoiceNote(1, 42, 32, 1)
        End Select
        di% = StartSound()
    End If
```

```
End Sub

Sub PlaySegment(SecsPos&, SegLength&)

    'SecsPos& = Int(SecsPos& / 22)'========test only===take out=====

    VMode = "Seg"
    TmpTot = 0
    i = 0
    TmpTot = f_video.MCIWnd1(0).Length
    Do While SecsPos& >= TmpTot
        i = i + 1
        TmpTot = TmpTot + f_video.MCIWnd1(i).Length
    Loop
    TmpCurAVI = i
    If TmpCurAVI <> CurAVI Then
        f_video.MCIWnd1(CurAVI).Visible = False
        CurAVI = TmpCurAVI
        SetPrevAVITot
    End If
    f_video.MCIWnd1(CurAVI).Position = SecsPos& - PrevAVITot
    DoEvents
    f_video.MCIWnd1(CurAVI).Visible = True
    StartPos& = SecsPos& - PrevAVITot
    EndPos& = StartPos& + SegLength&
    'f_video.Timer1.Enabled = True
    DoEvents
    If EndPos& <= f_video.MCIWnd1(CurAVI).Length Then
        f_video.MCIWnd1(CurAVI).Command = "Play From " & Format$(StartPos&) & " To " & Format$(E
ndPos&) & " Wait"
    Else
        f_video.MCIWnd1(CurAVI).Command = "Play From " & Format$(StartPos&) & " To " & Format$(f
_video.MCIWnd1(CurAVI).Length) & " Wait"
        NewSegLen& = EndPos& - f_video.MCIWnd1(CurAVI).Length
        If CancelFlag = True Then Exit Sub
        If CurAVI < NumAVIs - 1 Then
            f_video.MCIWnd1(CurAVI).Visible = False
            CurAVI = CurAVI + 1
            f_video.MCIWnd1(CurAVI).Visible = True
            f_video.MCIWnd1(CurAVI).Command = "Play From 0 To " & Format$(NewSegLen&) & " Wait"
        End If
    End If
    'f_video.Timer1.Enabled = False
    VMode = "Stop"
    DoEvents
End Sub

Sub QueryRun()
    Dim SArr() As String
    Dim Ds2 As snapshot
    Dim TAve  As Single
    Dim Fld1 As New field
    Dim Idx1 As New Index

    'On Error GoTo EH_RunQuery
    VMode = "Stop"
    f_video.MCIWnd1(CurAVI).Command = "Stop"
    'f_video.Timer1.Enabled = False
    If f_query.txtQuery.Text = "" Then
        Beep
        MsgBox "You must select one or more variables before you can run a query."
        Exit Sub
    End If
    screen.MousePointer = 11
    f_main.pnlStatus.Caption = "Running query ..."
    f_data.pnlGraph.Cls
    f_data.MaxPointer.Visible = False
    f_data.MaxPointer2.Visible = False
    f_data.MaxPointer3.Visible = False
    f_data.MinPointer.Visible = False
```

```
    f_data.MinPointer2.Visible = False
    f_data.MinPointer3.Visible = False
    f_data.LeadinPointer.Visible = False
    f_data.LeadoutPointer.Visible = False
    f_data.MaxPointer.Left = 375
    f_data.MaxPointer2.Left = 375
    f_data.MaxPointer3.Left = 375
    f_data.MinPointer.Left = 375
    f_data.MinPointer2.Left = 375
    f_data.MinPointer3.Left = 375
    f_data.LeadinPointer.Left = 375
    f_data.LeadoutPointer.Left = 375
    f_data.pnlSlider.Left = 435
    DoEvents
    QueryHasBeenRun = True

    'determine which variables are being used
    n% = 0
    For x% = 0 To f_query.List1.ListCount - 1
        If f_query.List1.Selected(x%) Then
            n% = n% + 1
        End If
    Next x%
    ReDim SArr(n%)
    n% = 0
    For x% = 0 To f_query.List1.ListCount - 1
        If f_query.List1.Selected(x%) Then
            SArr(n%) = f_query.List1.List(x%)
            n% = n% + 1
        End If
    Next x%

    'run query
    'TblVar.Index = "I_SECONDS"
    'TblVar.MoveLast
    'f_data.pnlGraph.ScaleWidth = TblVar("SECONDS")
    'BarInc = Int(f_data.pnlGraph.ScaleWidth / 10)
    'TblVar.MoveFirst
    x% = 0
    '======
    'if only one variable, find highs and lows and draw graph
    'directly from the database
    If n% = 1 Then
        NewFld$ = SArr(0)
    Else  'if more than one variable has been selected
        NewFld$ = ""
        For Y% = 0 To n% - 1
            NewFld$ = NewFld$ & SArr(Y%) & "/"
        Next Y%
        NewFld$ = UCase$(Left$(NewFld$, Len(NewFld$) - 1))
        fse% = -1
        For x% = 0 To f_query.List1.ListCount - 1
            If f_query.List1.List(x%) = NewFld$ Then fse% = x%
        Next x%
        If fse% = -1 Then
            f_main.pnlStatus.Caption = "Creating new variable " & NewFld$ & "..."
            f_query.List1.AddItem NewFld$
            f_query.List1.Refresh
            TblVar.Close
            Fld1.Name = NewFld$: Fld1.Type = DB_SINGLE
            Idx1.Name = "I_" & NewFld$: Idx1.Fields = NewFld$: Idx1.Primary = False: Idx1.Unique
= False
            NewDB.TableDefs("T_DATA").Fields.Append Fld1
            NewDB.TableDefs("T_DATA").Indexes.Append Idx1
            Set TblVar = NewDB.OpenTable("T_DATA")
            TblVar.MoveFirst
            TAve = 0
            Do While Not (TblVar.EOF)
                For Y% = 0 To n% - 1
                    TAve = TAve + TblVar(SArr(Y%))
```

```
                Next Y%
                TblVar.Edit
                TblVar(NewFld$) = TAve / n%
                TblVar.Update
                TAve = 0
                TblVar.MoveNext
            Loop
            TblVar.MoveFirst
        'Else
            'di% = SendMessage(f_query.List1.hWnd, LB_SELITEMRANGE, False, &HFFFF0000)
            'di% = SendMessage(f_query.List1.hWnd, LB_SELECTSTRING, -1, NewFld$)
        End If
    End If
    SQLST$ = "SELECT SECONDS, [" & NewFld$ & "] FROM T_DATA"
    Set Ds1 = NewDB.CreateSnapshot(SQLST$)

    '======Find the top 3 highs and lows
    f_main.pnlStatus.Caption = "Locating highs and lows for " & NewFld$ & " ..."
    If GetHisLosFromINI(NewFld$) = 0 Then FindHisLos NewFld$

    '======Draw the graph
    f_main.pnlStatus.Caption = "Drawing graph for " & NewFld$ & " ..."
    DrawGraph

    f_main.pnlStatus.Caption = "Query complete ..."

    'position AVI to x seconds before max and start playback
    DBIniFile$ = DBPath$ & Left$(DBFile$, Len(DBFile$) - 3) & "ini"
    SegLength% = GetPrivateProfileInt("Main", "AVISegLength", 0, DBIniFile$)
    SecsBefore% = GetPrivateProfileInt("Main", "AVISecsBefore", 0, DBIniFile$)
    GraphClick Int(MaxX1 - SecsBefore%)
    VideoPlay
    'PlaySegment (MaxX1 - SecsBefore%) * 1000, SegLength% * 1000

    Ds1.Close
    screen.MousePointer = 0
    Exit Sub
EH_RunQuery:
    If Err = 94 Then Resume Next Else MsgBox "Error in RunQuery:" & Error(Err)
    Exit Sub
End Sub

Sub SetPrevAVITot()
    PrevAVITot = 0
    If CurAVI > 0 Then
        For x = 0 To CurAVI - 1
            PrevAVITot = PrevAVITot + f_video.MCIWnd1(x).Length
        Next x
    End If
End Sub

Function StripPath$(T$)
    Dim x%, ct%
    StripPath$ = T$
    x% = InStr(T$, "\")
    Do While x%
        ct% = x%
        x% = InStr(ct% + 1, T$, "\")
    Loop
    If ct% > 0 Then StripPath$ = Mid$(T$, ct% + 1)
End Function

Sub TileBitmapOnForm(theForm As Form, BitmapCtl As PictureBox)
    Dim FormOrigScaleMode%, CtlOrigScaleMode%
    FormOrigScaleMode% = theForm.ScaleMode
    CtlOrigScaleMode% = BitmapCtl.ScaleMode
    theForm.ScaleMode = PIXELS
    BitmapCtl.ScaleMode = PIXELS
    Call TileBitmapOnFormEx(theForm, BitmapCtl, 0, 0, (theForm.ScaleWidth), (theForm.ScaleHeight
))
```

```
    theForm.ScaleMode = FormOrigScaleMode%
    BitmapCtl.ScaleMode = CtlOrigScaleMode%
End Sub

Sub TileBitmapOnFormEx(theForm As Form, BitmapCtl As PictureBox, xLeft%, yTop%, xRight%, yBot' n
%)
    Dim FormOrigScaleMode%, CtlOrigScaleMode%
    Dim BmpWidth%, BmpHeight%
    Dim DestWidth%, DestHeight%
    Dim BltWidth%, BltHeight%
    Dim CurXPos%, CurYPos%
    Dim bltOK%
    FormOrigScaleMode% = theForm.ScaleMode
    CtlOrigScaleMode% = BitmapCtl.ScaleMode
    theForm.ScaleMode = PIXELS
    BitmapCtl.ScaleMode = PIXELS
    BmpWidth% = BitmapCtl.ScaleWidth
    BmpHeight% = BitmapCtl.ScaleHeight
    DestWidth% = xRight% - xLeft%
    DestHeight% = yBottom% - yTop%
    BitmapCtl.Visible = True  '============
    For CurYPos% = xLeft% To DestHeight% Step BmpHeight%
        If (CurYPos% + BmpHeight%) <= DestHeight% Then
            BltHeight% = BmpHeight%
        Else
            BltHeight% = DestHeight% - CurYPos%
        End If
        For CurXPos% = yTop% To DestWidth% Step BmpWidth%
            If (CurXPos% + BmpWidth%) <= DestWidth% Then
                BltWidth% = BmpWidth%
            Else
                BltWidth% = DestWidth% - CurXPos%
            End If
            bltOK% = BitBLt(theForm.hDC, CurXPos%, CurYPos%, BltWidth%, BltHeight%, BitmapCtl  )
C, 0, 0, SRCCOPY)
            If bltOK% = False Then
                MsgBox "BitBlt failed in TileBitmapOnForm"
            End If
        Next CurXPos%
    Next CurYPos%
    theForm.ScaleMode = FormOrigScalMode%
    BitmapCtl.ScaleMode = CtlOrigScaleMode%
End Sub

Sub VideoEnd()
    VMode = "Stop"
    f_video.MCIWnd1(CurAVI).Command = "Stop"
    DoEvents
    f_main.pnlStatus.Caption = "Video advanced to end"
    If CurAVI <> NumAVIs - 1 Then
        f_video.MCIWnd1(CurAVI).Visible = False
        CurAVI = NumAVIs - 1
        SetPrevAVITot
        f_video.MCIWnd1(CurAVI).Visible = True
    End If
    f_video.MCIWnd1(CurAVI).Position = f_video.MCIWnd1(CurAVI).Length
    f_data.pnlSlider.Left = f_data.pnlResponse.Width - 315
End Sub

Sub VideoPlay()
    VMode = "Play"
    f_video.MCIWnd1(CurAVI).Command = "Play"
End Sub

Sub VideoRewind()
    VMode = "Stop"
    f_video.MCIWnd1(CurAVI).Command = "Stop"
    DoEvents
    f_main.pnlStatus.Caption = "Video rewound to beginning"
    If CurAVI <> 0 Then
```

```
        f_video.MCIWnd1(CurAVI).Visible = False
        CurAVI = 0
        SetPrevAVITot
        f_video.MCIWnd1(0).Visible = True
    End If
    f_video.MCIWnd1(0).Position = 0
    f_data.pnlSlider.Left = 435
End Sub

Sub VideoStop()
    VMode = "Stop"
    f_video.MCIWnd1(CurAVI).Command = "Stop"
End Sub

Sub WaitSecs(secs)
Dim Start!, temp%
    Start! = Timer
    While Timer < Start! + secs + 1
        temp% = DoEvents()
    Wend
End Sub

Sub WinNotOnTop(F As Form)
    wFlags% = SWP_NOMOVE Or SWP_NOSIZE
    Call SetWindowPos(F.hWnd, HWND_NOTOPMOST, 0, 0, 0, 0, wFlags%)
End Sub

Sub winOnTop(F As Form)
    wFlags% = SWP_NOMOVE Or SWP_NOSIZE
    Call SetWindowPos(F.hWnd, -1, 0, 0, 0, 0, wFlags%)
End Sub
```

```
F_VIDEO - 1


Private Sub cmdCancel_Click()
    CancelFlag = True
    'f_video.MCIWnd1(CurAVI).Command = "Stop"
End Sub

Private Sub cmdEnd_Click()
    VideoEnd
End Sub

Private Sub cmdPlay_Click()
    VideoPlay
End Sub

Private Sub cmdRewind_Click()
    VideoRewind
End Sub

Private Sub cmdStop_Click()
    VideoStop
End Sub

Private Sub cmdVolume_Click(index As Integer)
    cmdVolume(index).Enabled = False
    For x = 0 To NumAVIs - 1
        NV% = F_VIDEO.MCIWnd1(x).Volume + Choose(index + 1, 100, -100)
        If NV% > 0 Then F_VIDEO.MCIWnd1(x).Volume = NV%
    Next x
    DoEvents
    cmdVolume(index).Enabled = True
End Sub

Private Sub Form_Load()
    'if there are settings for form placement, use them, else ...
    Me.Move 0, 0, F_QUERY.Left, F_DATA.Top
End Sub

Private Sub Form_Resize()
    DoEvents
    PositionControls
End Sub

Private Sub MCIWnd1_positionchange(index As Integer, Position As Long)
    T% = Int((Position + PrevAVITot) / 1000)
    Vid$ = Format$(Int(T% / 60), "00") & ":" & Format$(T% Mod 60, "00")
    If VMode = "Play" Then
        If Position = MCIWnd1(CurAVI).Length Then
            If CurAVI < NumAVIs - 1 Then
                'MCIWnd1(CurAVI).Command = "Stop"
                MCIWnd1(CurAVI).Visible = False
                CurAVI = CurAVI + 1
                MCIWnd1(CurAVI).Visible = True
                MCIWnd1(CurAVI).Position = 0
                MCIWnd1(CurAVI).Command = "Play"
                SetPrevAVITot
            Else
                VMode = "Stop"
            End If
        Else
            F_MAIN.pnlStatus.Caption = "Video playing at " & Vid$
            q% = Int((Position + PrevAVITot) / AVILength * F_DATA.pnlGraph.Width)
            If q% < 0 Then q% = 0
            F_DATA.pnlSlider.Left = 435 + q%
            DoEvents
        End If
    ElseIf VMode = "Stop" Then
        F_MAIN.pnlStatus.Caption = "Video stopped at " & Vid$
    Else
        F_MAIN.pnlStatus.Caption = "Video playing at " & Vid$
        q% = Int((Position + PrevAVITot) / AVILength * F_DATA.pnlGraph.Width)
```

```
F_VIDEO - 2

        If q% < 0 Then q% = 0
        F_DATA.pnlSlider.Left = 435 + q%
        DoEvents
    End If
End Sub

Private Sub PositionControls()
    If MCIWnd1(0).Filename <> "" Then
        WR1 = F_VIDEO.MCIWnd1(0).Width / (F_VIDEO.Width - 345)
        HR1 = F_VIDEO.MCIWnd1(0).Height / (F_VIDEO.Height - 1220)
        If WR1 > HR1 Then
            F_VIDEO.MCIWnd1(0).Zoom = F_VIDEO.MCIWnd1(0).Zoom / WR1
        Else
            F_VIDEO.MCIWnd1(0).Zoom = F_VIDEO.MCIWnd1(0).Zoom / HR1
        End If
    End If
    If FormLoaded("F_SYNOP") Then
        pnlVidFrame.Move 0, 300, F_VIDEO.MCIWnd1(0).Width + 240, F_VIDEO.MCIWnd1(0).Height + 240
    Else
        pnlVidFrame.Move 0, 0, F_VIDEO.MCIWnd1(0).Width + 240, F_VIDEO.MCIWnd1(0).Height + 240
    End If
    For x = 1 To NumAVIs - 1
        F_VIDEO.MCIWnd1(x).Zoom = F_VIDEO.MCIWnd1(0).Zoom
    Next x
    'MsgBox Format$(f_video.MCIWnd1(0).Zoom)
End Sub

Private Sub Spin1_SpinDown()
    p& = F_VIDEO.MCIWnd1(CurAVI).Position - 5000

    If p& < 0 Then
        If CurAVI > 0 Then
            F_VIDEO.MCIWnd1(CurAVI).Visible = False
            CurAVI = CurAVI - 1
            SetPrevAVITot
            F_VIDEO.MCIWnd1(CurAVI).Visible = True
            F_VIDEO.MCIWnd1(CurAVI).Position = F_VIDEO.MCIWnd1(CurAVI).Length + p&
            p& = F_VIDEO.MCIWnd1(CurAVI).Position
        Else
            VMode = "Stop"
            p& = 0
        End If
    Else
        F_VIDEO.MCIWnd1(CurAVI).Position = p&
    End If
    'F_Main.pnlStatus.Caption = "Video playing at " & Int((p& + PrevAVITot) / 1000)
    F_DATA.pnlSlider.Left = 435 + (p& + PrevAVITot) / AVILength * F_DATA.pnlGraph.Width
    'DoEvents
End Sub

Private Sub Spin1_SpinUp()
    p& = F_VIDEO.MCIWnd1(CurAVI).Position + 5000

    If p& > F_VIDEO.MCIWnd1(CurAVI).Length Then
        If CurAVI < NumAVIs - 1 Then
            p& = p& - F_VIDEO.MCIWnd1(CurAVI).Length
            F_VIDEO.MCIWnd1(CurAVI).Visible = False
            CurAVI = CurAVI + 1
            SetPrevAVITot
            F_VIDEO.MCIWnd1(CurAVI).Visible = True
            F_VIDEO.MCIWnd1(CurAVI).Position = 0
        Else
            VMode = "Stop"
            p& = F_VIDEO.MCIWnd1(CurAVI).Length
        End If
    Else
        F_VIDEO.MCIWnd1(CurAVI).Position = p&
    End If
    'F_Main.pnlStatus.Caption = "Video playing at " & Int((p& + PrevAVITot) / 1000)
    F_DATA.pnlSlider.Left = 435 + (p& + PrevAVITot) / AVILength * F_DATA.pnlGraph.Width
```

```
F_VIDEO - 3

    'DoEvents

End Sub

Private Sub Timer1_Timer()
    p& = F_VIDEO.MCIWnd1(CurAVI).Position
    T% = Int((p& + PrevAVITot) / 1000)
    Vid$ = Format$(Int(T% / 60), "00") & ":" & Format$(T% Mod 60, "00")
    F_MAIN.pnlStatus.Caption = "Video playing at " & Vid$
    q% = Int((p& + PrevAVITot) / AVILength * F_DATA.pnlGraph.Width)
    If q% < 0 Then q% = 0
    F_DATA.pnlSlider.Left = 435 + q%
    DoEvents
End Sub
```

```
F_SYNOP - 1

Dim CurList1 As Integer
Dim DY
Dim TmpDesc$

Private Sub chkMax1_Click()
    If chkMax1.Value = 0 Then chkMax2.Value = 0: chkMax3.Value = 0
End Sub

Private Sub chkMax2_Click()
    If chkMax2.Value = 1 Then chkMax1.Value = 1 Else chkMax3.Value = 0
End Sub

Private Sub chkMax3_Click()
    If chkMax3.Value = 1 Then chkMax1.Value = 1: chkMax2.Value = 1
End Sub

Private Sub chkMin1_Click()
    If chkMin1.Value = 0 Then chkMin2.Value = 0: chkMin3.Value = 0
End Sub

Private Sub chkMin2_Click()
    If chkMin2.Value = 1 Then chkMin1.Value = 1 Else chkMax3.Value = 0
End Sub

Private Sub chkMin3_Click()
    If chkMin3.Value = 1 Then
        chkMin1.Value = 1
        chkMin2.Value = 1
    End If
End Sub

Private Sub cmdAdd_Click()
    If List3.ListCount > 0 Then
        List3.ZOrder 0
        List3.ListIndex = 0
        List3.Visible = True
    Else
        Beep
    End If
End Sub

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdDelete_Click()
    cmdDelete.Enabled = False
    n% = List1.ListIndex
    T$ = List1.List(n%)
    List3.AddItem T$
    List1.RemoveItem n%
    List2.RemoveItem n%
    DoEvents
    If n% > List1.ListCount - 1 Then
        CurList1 = List1.ListCount - 1
        List1.ListIndex = List1.ListCount - 1
        List2.ListIndex = List2.ListCount - 1
    Else
        CurList1 = n%
        List1.ListIndex = n%
        List2.ListIndex = n%
    End If
    cmdDelete.Enabled = True
End Sub

Private Sub cmdDown_Click()
    cmdDown.Enabled = False
    n% = List1.ListIndex
    T$ = List1.List(n%)
    t2$ = List2.List(n%)
```

```
    If n% < List1.ListCount - 1 Then
        CurList1 = n% + 1
        List1.RemoveItem n%
        List1.AddItem T$, n% + 1
        List2.RemoveItem n%
        List2.AddItem t2$, n% + 1
        List1.ListIndex = n% + 1
    End If
    cmdDown.Enabled = True
End Sub

Private Sub cmdOK_Click()
    Dim VH%, VW%
    CancelFlag = False
    If chkMax1.Value = 0 And chkMin1.Value = 0 Then
        MsgBox "You must select one of the max/min check boxes."
        Exit Sub
    ElseIf txtSegLength.Text = "" Or txtSecsBefore.Text = "" Then
        MsgBox "You must have values in both Segment Length and Start."
        Exit Sub
    End If
    di% = WritePrivateProfileString("Main", "SynopSegLength", txtSegLength.Text, DBIniFile$)
    di% = WritePrivateProfileString("Main", "SynopSecsBefore", txtSecsBefore.Text, DBIniFile$)

    F_SYNOP.Hide
    DoEvents
    VH% = F_VIDEO.Height
    VW% = F_VIDEO.Width
    F_VIDEO.MCIWnd1(CurAVI).Visible = False
    F_VIDEO.pnlVidFrame.Visible = False
    F_VIDEO.WindowState = 2
    DoEvents
    F_MAIN.pnlStatus.Caption = "Press ESC to stop video output"
    F_VIDEO.Caption = txtMainTitle.Text
    F_VIDEO.pnlVidTools.Visible = False
    'f_video.lblDesc.Move f_video.pnlVidOut.Width - 4000
    F_VIDEO.lblVariable.Caption = ""
    F_VIDEO.cmdCancel.Left = F_VIDEO.pnlVidOut.Width - 1170
    F_VIDEO.pnlVidOut.Visible = True
    F_VIDEO.FXMainTitle.Width = F_VIDEO.pnlVidOut.Width - 600
    F_VIDEO.FXMainTitle.Caption = txtMainTitle.Text
    F_VIDEO.FXMainTitle.Visible = True
    DoEvents
    Beep
    MsgBox "Hit Enter and set your VCR to RECORD"
    WaitSecs 10
    If CancelFlag = True Then GoTo EH_VidOut1
    F_VIDEO.FXMainTitle.Visible = False
    F_VIDEO.pnlVidFrame.Visible = True
    SegLength& = Val(txtSegLength.Text) * 1000
    'for each variable in list, print header, high (and low)
    For X = 0 To List1.ListCount - 1
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        VNam$ = List1.List(X)
        If GetHisLosFromINI(VNam$) = 0 Then
            SQLST$ = "SELECT SECONDS, [" & VNam$ & "] FROM T_DATA"
            Set Ds1 = NewDB.CreateSnapshot(SQLST$)
            FindHisLos VNam$
            Ds1.Close
        End If
        VDesc$ = List2.List(X)
        If VDesc$ = "@" Then VDesc$ = ""
        F_VIDEO.lblDesc.Caption = VDesc$
        DoEvents
        If chkMax1.Value = 1 Then
            SegStart& = (MaxX1 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "High 1: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
```

```
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        If chkMax2.Value = 1 Then
            SegStart& = (MaxX2 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "High 2: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        If chkMax3.Value = 1 Then
            SegStart& = (MaxX3 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "High 3: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        If chkMin1.Value = 1 Then
            SegStart& = (MinX1 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "Low 1: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        If chkMin2.Value = 1 Then
            SegStart& = (MinX2 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "Low 2: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
        DoEvents
        If CancelFlag = True Then GoTo EH_VidOut1
        If chkMin3.Value = 1 Then
            SegStart& = (MinX3 - Val(txtSecsBefore.Text)) * 1000
            F_VIDEO.lblVariable.Caption = "Low 3: " & VNam$
            PlaySegment SegStart&, SegLength&
        End If
    Next X
    F_VIDEO.pnlVidFrame.Visible = False
    WaitSecs 4
    Beep
    F_MAIN.pnlStatus.Caption = "Recording complete ...  Stop tape"
    GoTo EH_VidOut2
EH_VidOut1:
    Beep
    Beep
    F_MAIN.pnlStatus.Caption = "Recording canceled ...  Stop tape"
EH_VidOut2:
    Unload F_SYNOP
    F_VIDEO.Caption = "Video"
    F_VIDEO.FXMainTitle.Visible = False
    F_VIDEO.pnlVidOut.Visible = False
    F_VIDEO.pnlVidTools.Visible = True
    F_VIDEO.pnlVidFrame.Move 0, 0
    F_VIDEO.WindowState = 0
    F_VIDEO.pnlVidFrame.Visible = True
    DoEvents
    Exit Sub
End Sub

Private Sub cmdUp_Click()
    cmdUp.Enabled = False
    n% = List1.ListIndex
    T$ = List1.List(n%)
    t2$ = List2.List(n%)
    If n% > 0 Then
        CurList1 = n% - 1
        List1.RemoveItem n%
        List1.AddItem T$, n% - 1
        List2.RemoveItem n%
        List2.AddItem t2$, n% - 1
        List1.ListIndex = n% - 1
```

```
    End If
    cmdUp.Enabled = True
End Sub

Private Sub Form_Load()
    Dim RtnString As String * 8
    centerform Me
    winOnTop Me
    txtSegLength.Text = Format$(GetPrivateProfileInt("Main", "SynopSegLength", 0, DBIniFile$))
    txtSecsBefore.Text = Format$(GetPrivateProfileInt("Main", "SynopSecsBefore", 0, DBIniFile$))
    For X = 0 To F_QUERY.List1.ListCount - 1
        List1.AddItem F_QUERY.List1.List(X)
    Next X
    For X = 0 To F_QUERY.List1.ListCount - 1
        List2.AddItem "@"
    Next X
    List1.ListIndex = 0
    List2.ListIndex = 0
    CurList1 = 0
End Sub

Private Sub List1_click()
    'List2.List(CurList1) = txtVarDesc
    L2$ = List2.List(List1.ListIndex)
    If L2$ = "@" Then txtVarDesc = "" Else txtVarDesc = L2$
    CurList1 = List1.ListIndex
End Sub

Private Sub List1_DragDrop(Source As Control, X As Single, Y As Single)
    'Label12.Visible = False
    'Beep
End Sub

Private Sub List1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
    'Label12.Move List1.Left, List1.Top + Y - DY / 2
End Sub

Private Sub List1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 Then
        DY = TextHeight("A")     ' Get height of one line.
        n% = Int(Y / DY)
        If n% > List1.ListCount - 1 Then Exit Sub
        Label12.Caption = List1.List(n%)
        TmpDesc$ = List2.List(n%)
        List1.ListIndex = -1
        List1.RemoveItem n%
        List2.RemoveItem n%
        Label12.Move List1.Left, List1.Top + Y - DY / 2, List1.Width
        Label12.ZOrder 0
        Label12.Visible = True
    End If
End Sub

Private Sub List1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 And Y > 0 And Y < List1.Height And Label12.Visible = True Then
        Label12.Move List1.Left, List1.Top + Y - DY / 2
    End If
End Sub

Private Sub List1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 And Label12.Visible = True Then
        Label12.Visible = False
        If Y < 0 Then
            Y = 0
        ElseIf Y > List1.Height Then
            Y = List1.Height
        End If
        Newx = Int(Y / DY)
        If Newx > List1.ListCount - 1 Then Newx = List1.ListCount
        List1.AddItem Label12.Caption, Newx
```

```
        List2.AddItem TmpDesc$, Newx
        List1.ListIndex = Newx
    End If
End Sub

Private Sub List3_DblClick()
    n% = List3.ListIndex
    List1.AddItem List3.List(n%)
    List2.AddItem "@"
    List3.Visible = False
    List3.RemoveItem n%
End Sub

Private Sub TapePrintMax(VNam$)

End Sub

Private Sub TapePrintMin(VNam$)

End Sub

Private Sub TapePrintTitle(TT As String)

End Sub

Private Sub txtVarDesc_LostFocus()
    List2.List(CurList1) = txtVarDesc
End Sub
```

```
F_QUERY - 1


Private Sub cmdDelete_Click()
    'make sure that only one line is highlighted
    n% = 0
    For X% = 0 To F_QUERY.List1.ListCount - 1
        If F_QUERY.List1.Selected(X%) Then
            n% = n% + 1
        End If
    Next X%
    If n% = 0 Then
        Exit Sub
    ElseIf n% > 1 Then
        Beep
        MsgBox "You can only delete variables one at a time."
        Exit Sub
    End If
    'confirm the delete
    Msg$ = "Are you sure you want to delete the variable " & F_QUERY.List1.List(ListIndex) & " ?
"
    Msg$ = Msg$ & Chr(10) & "This will delete the data for that variable from the database. "
    Msg$ = Msg$ & "It cannot be retrieved."
    Beep
    If MsgBox(Msg$, 305, "Delete Variable") = 1 Then
        'remove item from list
        n% = List1.ListIndex
        List1.RemoveItem n%
        'create a new database and table with all but the selected field

        'copy over the data to the new database

        'delete the original database

        'rename the new database

    End If
End Sub

Private Sub cmdRunQuery_Click()
    F_QUERY.cmdRunQuery.Enabled = False
    QueryRun
    F_QUERY.cmdRunQuery.Enabled = True
End Sub

Private Sub Form_Load()
    'if there are settings for form placement, use them, else ...
    Me.Move screen.Width * 0.75, 0, screen.Width / 4, screen.Height * 0.65
End Sub

Private Sub Form_Resize()
    PositionControls
End Sub

Private Sub List1_click()
    n = 1
    For X = 0 To List1.ListCount - 1
        If List1.Selected(X) Then
            If n = 1 Then
                T$ = List1.List(X)
            Else
                T$ = T$ + " AND " & List1.List(X)
            End If
            n = n + 1
        End If
    Next X
    txtQuery = T$
End Sub

Private Sub List1_DblClick()
    'f_query.cmdRunQuery.Enabled = False
    'QueryRun
```

```
    'f_query.cmdRunQuery.Enabled = True
End Sub

Private Sub PositionControls()
    'read width, height, left, top from ini

    pnlQuery.Move 0, 0, Me.Width - 105, Me.Height - 390
    List1.Move 180, 180, pnlQuery.Width - 360, pnlQuery.Height - 1500
    txtQuery.Move 180, List1.Top + List1.Height + 105, List1.Width, 795
    cmdRunQuery.Move 180, txtQuery.Top + 840, List1.Width - 420, 315
    cmdDelete.Move cmdRunQuery.Width + 225, cmdRunQuery.Top
End Sub
```

```
F_PREFS - 1

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdRecalc_Click()
    For m = 0 To F_QUERY.List1.ListCount - 1
        NewFld$ = UCase$(F_QUERY.List1.List(m))
        SQLST$ = "SELECT SECONDS, [" & NewFld$ & "] FROM T_DATA"
        Set Ds1 = NewDB.CreateSnapshot(SQLST$)
        F_MAIN.pnlStatus.Caption = "Recalculating highs and lows for " & NewFld$
        FindHisLos NewFld$
        Ds1.Close
    Next m
    Beep
    F_MAIN.pnlStatus.Caption = "Recalculation complete"
End Sub

Private Sub cmdSave_Click()
    di% = WritePrivateProfileString("Main", "AVISegLength", txtSegLength.Text, DBIniFile$)
    di% = WritePrivateProfileString("Main", "AVISecsBefore", txtSecsBefore.Text, DBIniFile$)
    Unload Me
End Sub

Private Sub Form_Load()
    centerform Me
    txtSegLength.Text = Format$(GetPrivateProfileInt("Main", "AVISegLength", 0, DBIniFile$))
    txtSecsBefore.Text = Format$(GetPrivateProfileInt("Main", "AVISecsBefore", 0, DBIniFile$))
    txtBegExclude.Text = Format$(GetPrivateProfileInt("Main", "AVILeadin", 0, DBIniFile$))
    txtEndExclude.Text = Format$(GetPrivateProfileInt("Main", "AVILeadout", 0, DBIniFile$))
End Sub

Private Sub frmVideoPrefs_Click()

End Sub

Private Sub txtBegExclude_LostFocus()
    di% = WritePrivateProfileString("Main", "AVILeadin", txtBegExclude.Text, DBIniFile$)
End Sub

Private Sub txtEndExclude_LostFocus()
    di% = WritePrivateProfileString("Main", "AVILeadout", txtEndExclude.Text, DBIniFile$)
End Sub
```

```
F_NEWDB - 1

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdOK_Click()
    Dim NewDBPath As String
    screen.MousePointer = 11
    DBN$ = txtDBName.Text
    If Exists(app.Path & "\" & DBN$) > 0 Then
        Msg$ = "A database exists by that name, enter another."
        Beep
        MsgBox Msg$, 0, "Database exists"
        txtDBName.SetFocus
        txtDBName.SelStart = 0
        txtDBName.SelLength = Len(DBN$)
        Exit Sub
    End If
    'create a new directory
    MkDir app.Path & "\" & DBN$
    F_NEWDB.Hide
    DoEvents
    'create a new .ini file
    DBIniFile$ = app.Path & "\" & DBN$ & "\" & DBN$ & ".ini"
    Open DBIniFile$ For Output As #1
    Print #1, "[Main]"
    Print #1, "AVIPath=" & txtVidFilter.Text & ".AVI"
    Print #1, "AVIOverlap="
    Print #1, "AVILength="
    Print #1, "AVILeadin=0"
    Print #1, "AVILeadout=0"
    Print #1, "AVIDrives=" & UCase$(txtVidFileDrives.Text)
    Print #1, "AVISegLength=20"
    Print #1, "AVISecsBefore=10"
    Print #1, "SynopSegLength=20"
    Print #1, "SynopSecsBefore=10"
    Close
    'create the new database
    F_MAIN.pnlStatus.Caption = "Importing DIF files"
    NewDBPath = app.Path & "\" & DBN$ & "\" & DBN$ & ".mdb"
    DD$ = Dir1.Path
    If Right$(DD$, 1) <> "\" Then DD$ = DD$ & "\"
    TT$ = DD$ & File1.Filename
    ImportDIF TT$, NewDBPath
    Unload F_NEWDB
    DoEvents
    'open the new database
    F_MAIN.pnlStatus.Caption = "Opening new database"
    DBOpen NewDBPath
    Beep
    F_MAIN.pnlStatus.Caption = "New database creation complete"
    screen.MousePointer = 0
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Form_Load()
    centerform Me
    winOnTop Me
End Sub

Private Sub frmVidFiles_Click()

End Sub
```

```
F_MAIN - 1

Private Sub MDIForm_Load()
    mnuHelp.Caption = Chr(8) & mnuHelp.Caption
    QueryHasBeenRun = False
    DBPath$ = ""
    DoEvents
    Load F_QUERY
    Load F_DATA
    Load F_VIDEO
    DoEvents
End Sub

Private Sub mnuFileExit_Click()
    VideoStop
    Beep
    If MsgBox("Do you want to quit?", 292, "Quit?") = 6 Then
        If Len(DBPath$) Then
            screen.MousePointer = 11
            TblVar.Close
            NewDB.Close
            'f_main.pnlStatus.Caption = "Compacting database file ..."
            'DBCompact CurDBPath
            screen.MousePointer = 0
        End If
        End
    End If
End Sub

Private Sub mnuFileImportDIF_Click()
    'ImportDIF
End Sub

Private Sub mnuFileNew_Click()
    F_NEWDB.Show
End Sub

Private Sub mnuFileOpen_Click()
    'locate the .mdb
    On Error GoTo EH_OpenCancel
    F_QUERY.CMDialog1.DialogTitle = "Open Database"
    F_QUERY.CMDialog1.CancelError = True
    F_QUERY.CMDialog1.Filter = "Database file|*.mdb"
    F_QUERY.CMDialog1.FilterIndex = 1
    F_QUERY.CMDialog1.Action = 1
    CurDBPath = F_QUERY.CMDialog1.Filename

    On Error GoTo EH_OpenDatabase
    screen.MousePointer = 11
    DBOpen CurDBPath
    screen.MousePointer = 0
    Exit Sub
EH_OpenCancel:

    Exit Sub
EH_OpenDatabase:
    screen.MousePointer = 0
    MsgBox "Error OpenDatabase:" & Error(Err)
    Exit Sub

End Sub

Private Sub mnuFilePrefs_Click()
    F_PREFS.Show 1
End Sub

Private Sub mnuFileSynopsis_Click()
    F_SYNOP.Show
End Sub

Private Sub mnuHelpAbout_Click()
    Load F_ABOUT
```

```
    F_ABOUT.Timer2.Enabled = True
    F_ABOUT.Show
End Sub

Private Sub mnuVideoPlay_Click()
    VideoPlay
End Sub

Private Sub mnuVideoRewind_Click()
    VideoRewind
End Sub

Private Sub mnuVideoStop_Click()
    VideoStop
End Sub
```

```
F_DATA - 1


Private Sub Form_Deactivate()

End Sub

Private Sub Form_Initialize()

End Sub


Private Sub Form_Load()
    'if there are settings for form placement, use them, else ...
    Me.Move 0, screen.Height * 0.8 - 840, screen.Width, screen.Height * 0.2
End Sub

Private Sub Form_LostFocus()

End Sub


Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

End Sub


Private Sub Form_Resize()
    PositionControls
    'if a query has been run, redraw query graph
    If QueryHasBeenRun Then
        SQLST$ = "SELECT SECONDS, [" & NewFld$ & "] FROM T_DATA"
        Set Ds1 = NewDB.CreateSnapshot(SQLST$)
        DrawGraph
        Ds1.Close
    End If
End Sub

Private Sub pnlGraph_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 And VMode <> "Play" Then
        pnlSlider.Left = 435 + Int(X / pnlGraph.ScaleWidth * pnlGraph.Width)
    End If
End Sub

Private Sub pnlGraph_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 And VMode <> "Play" Then
        pnlSlider.Left = 435 + Int(X / pnlGraph.ScaleWidth * pnlGraph.Width)
    End If

End Sub

Private Sub pnlGraph_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    GraphClick X
End Sub

Private Sub PositionControls()
    pnlResponse.Move 0, 0, Me.Width - 105, Me.Height - 105
    pnlGraph.Move 480, 240, pnlResponse.Width - 750, pnlResponse.Height - 750
    Label3.Top = pnlGraph.Height + 165
    MinPointer.Top = pnlGraph.Height + 255
    MinPointer2.Top = pnlGraph.Height + 255
    MinPointer3.Top = pnlGraph.Height + 255
    MinPointer.ZOrder 1
    pnlSlider.Height = pnlGraph.Height
    For X = 0 To 9
        GSecs(X).Top = MinPointer.Top + 30
    Next X
    If GSecs(0).Caption <> "" Then AddGraphSeconds
    pnlGraph.Cls
    inc1 = pnlGraph.ScaleWidth / 10
    inc2 = pnlGraph.ScaleHeight / 4
    For X = 0 To 8
```

```
        Line1(X).X1 = inc1 * (X + 1)
        Line1(X).Y1 = 0
        Line1(X).X2 = inc1 * (X + 1)
        Line1(X).Y2 = pnlGraph.ScaleHeight
    Next X
    For X = 0 To 2
        Line2(X).X1 = 0
        Line2(X).Y1 = inc2 * (X + 1)
        Line2(X).X2 = pnlGraph.ScaleWidth
        Line2(X).Y2 = inc2 * (X + 1)
    Next X
End Sub

Private Sub pnlResponse_Click()

End Sub


Private Sub pnlResponse_GotFocus()

End Sub


Private Sub pnlResponse_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

End Sub

Private Sub pnlResponse_Paint()

End Sub


Private Sub pnlSlider_Click()

End Sub


Private Sub pnlSlider_DblClick()

End Sub

Private Sub pnlSlider_GotFocus()

End Sub
```

```
F_ABOUT - 1

Dim CurCell As Integer

Private Sub Form_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    winOnTop Me
    Picture1.Picture = PicClip1.GraphicCell(0)
    CurCell = 0
End Sub

Private Sub FXLabel1_Click()

End Sub

Private Sub FXLabel1_DblClick()

End Sub


Private Sub Label1_Click()

End Sub


Private Sub PicClip1_Click()

End Sub


Private Sub PicClip1_Paint()

End Sub '


Private Sub Timer1_Timer()
    txtInit.Visible = Not (txtInit.Visible)
    If CurCell = 0 Then CurCell = 1 Else CurCell = 0
    Picture1.Picture = PicClip1.GraphicCell(CurCell)
End Sub

Private Sub Timer2_Timer()
    Unload Me
End Sub
```

Ex. C

Maguire Project - PR Document

[redacted]

Introduction

Picture the typical focus group from the perspective of a participant. You are a member of an audience which has been assembled to watch a taped or live presentation, and you are asked to provide your moment-to-moment opinions. After the presentation, there may be a discussion period, which you are also asked to score. The job is relatively simple - give your honest responses and opinions.

Now look at it from the perspective of the presenters. You need to understand who the audience is, both as a group and individually. What are the age ranges, the income brackets, and the gender breakdown? What initial opinions and biases does the audience bring with them (and how have you swayed them from those opinions)? You need to time-match precisely the responses from the audience with the presentation. And you need to be able to analyze the results in a number of different ways. Analysis *is*, after all, the purpose of holding the focus group in the first place.

There are a number of technical challenges facing the presenters when it comes to data gathering and analysis. There are systems available today that address the data gathering end of the process, but analysis is still relatively unsophisticated and labor-intensive. This is where the Automated Video Analysis System (AVAS) comes in.

The AVAS System in Action

The goal of the AVAS system is to make it easier to analyze the results of a focus group session. As soon as the presentation is over, the data is ready for analysis. And the client is able to walk away with a video tape summary of the results just minutes later.

Here, briefly, is how the system works:

Video Digitizing: One of the key elements of the system is its ability to store the presentation on the computer in a digital format. The system includes the hardware and software needed to capture the presentation video in real time. Converting to a digital form allows the user to match responses to precise points in the video, quickly moving to any portion of the video and controlling its playback.

Data Importing: The system interfaces with existing audience data gathering processes and is flexible enough to accommodate a more integrated system in the future. It imports audience response data (by variable, over time) and stores it in its own indexed database.

Max/Min Analysis: When the data import is complete, a Max/Min analysis is performed on each identified variable (e.g.; male, female, over 50). The top three and bottom three points are calculated and stored for each of these variables. The user can exclude portions of the presentation from this analysis; for example, the first minute and last two minutes may be unimportant.

The Analysis Interface: The product has a simple interface, as shown in Figure 1. The user can setup a new database for a presentation or open any past database that has been created. There are three windows: a Query window that contains the list of variables, a Video window in which the digitized video is shown, and a Response window that represents the presentation timeline, shows a graph of responses for a variable, and identifies its high and low points.

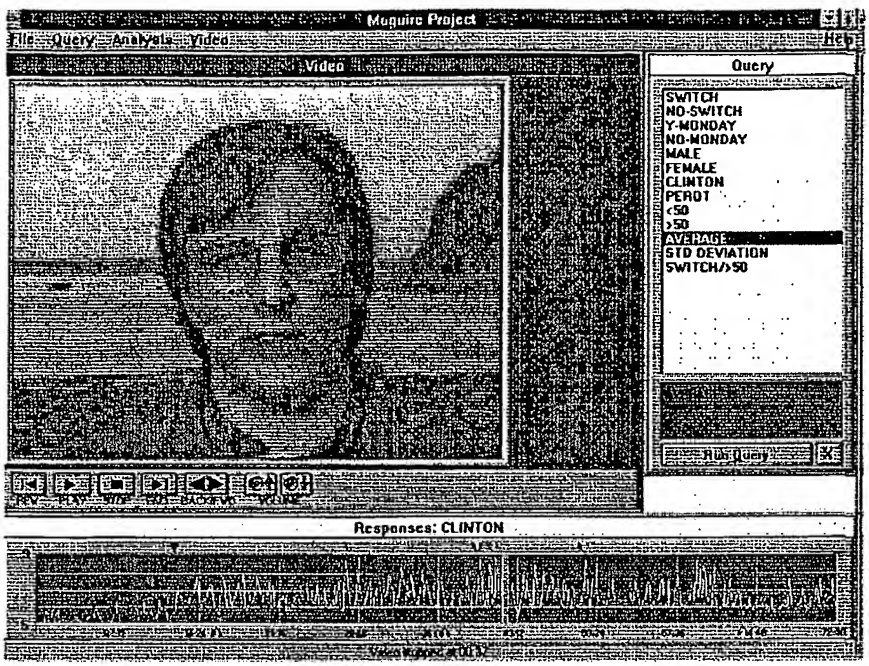


Figure 1 - Analysis Interface

Running a Query: To perform a query, the user selects one or more variables from the list and clicks Run Query. By selecting more than one variable, the user is, in effect, creating a new variable; e.g.; selecting "Male" and "> 50" creates the new variable "Male \ > 50", which is then listed as a separate variable. As the query runs, the responses are graphed over time and the highs and lows are indicated. When the graph is complete, a video clip representing the highest response for that variable is automatically played in the Video window.

Freeform Browsing: The user can do freeform browsing and analysis of the responses. By clicking anywhere in the Response window timeline, the video moves to that point. The user controls video playback by means of VCR-like buttons and can also control the playback volume.

Summary Tape Creation: The user can create a customized summary tape containing video clips that represent the highs and lows for each variable, see Figure 2. The user can select the variables that are included and their sequence, the combinations of highs and lows, and the duration of clips. Descriptions of each variable can also be included. The summary is constructed and transferred to a local VCR connected to the system. The result is a concise video summary and analysis of the original presentation.

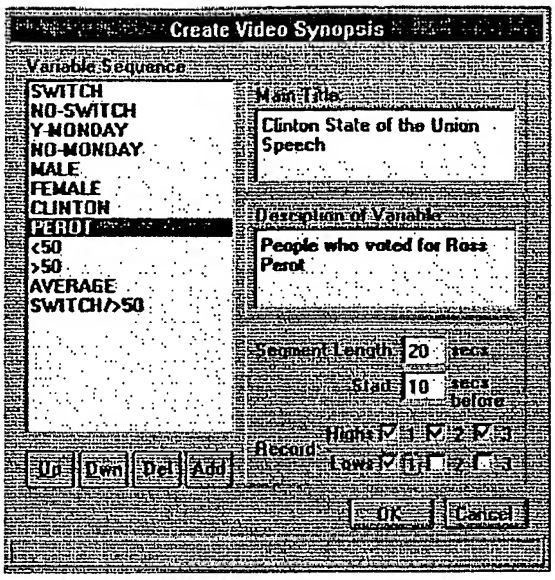


Figure 2 - Creating a Summary Tape

Future Directions

Among the areas being explored for future versions of the AVAS system are:

Integrated Data Gathering: We are looking at ways to better integrate the data gathering and analysis processes, so the results go directly into a database format rather than through the initial spreadsheet format. We are also looking into storing results on an individual basis, rather than just across pre-defined variables. This would give us an even more exact result for multivariable queries.

Better Data Exclusion: We want the user to be able to exclude any portion of data from the analysis. This will prevent random data-gathering faults from skewing the results.

Further Automation for Video Digitizing: We would like to make the process of digitizing video as automated as possible, which means better integration of the digitizing hardware and the video presentation system.

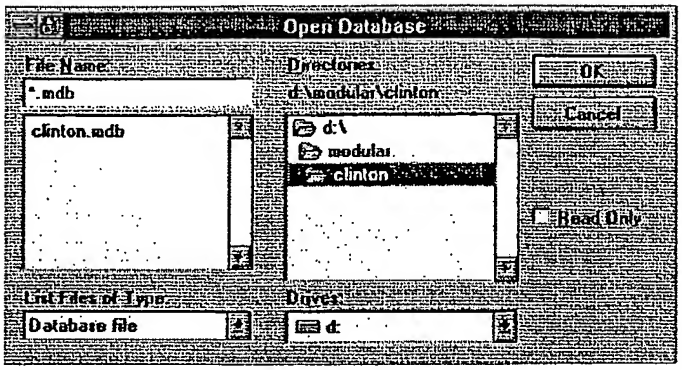


Figure 3 - Open Database

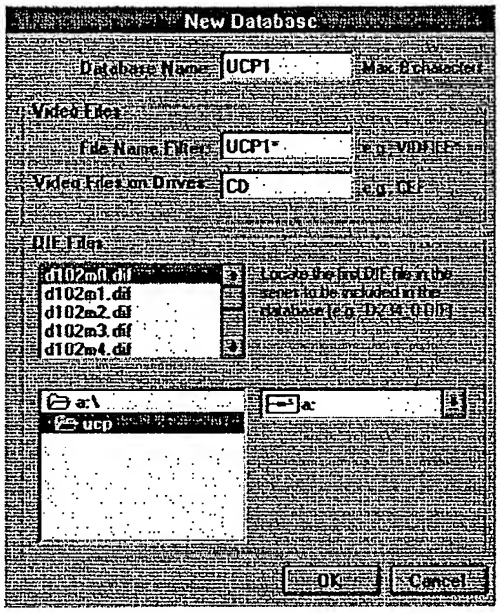


Figure 4- New Database